Preserving Privacy in Collaborative Business Process Composition

Hassaan Irshad[1], Basit Shafiq[1], Jaideep Vaidya[2], Muhammad Ahmed Bashir[1], Shafay Shamail[1] and Nabil Adam[3]

[1]*Department of Computer Science, Lahore University of Management Sciences, Lahore, Pakistan*

[2]*Management Science and Information Systems Department, Rutgers University, Newark, New Jersey, U.S.A.*

[3]*Rutgers Institute for Data Science, Learning, and Applications, Rutgers University, Newark, New Jersey, U.S.A.*

Keywords: Business Process Composition, Privacy.

Abstract: Collaborative business process composition exploits the knowledge of existing business processes of related organizations to compose an executable business process for a given organization based on its requirements and design specifications. Typically, this requires organizations to share and upload their existing business process execution sequences to a central repository. However, even after masking of confidential data, the execution sequences may still include sensitive business information which organizations may not want to share with their competitors. To address this issue, we develop a privacy-preserving Business Process Recommendation and Composition System (BPRCS), that generates a differentially private dataset of execution sequences which can be published and shared with other organizations for composition and implementation of their business processes. We also employ process mining and classification techniques on this differentially private dataset to regenerate the executable business process workflow. We experimentally validate the effectiveness of our approach.

1 INTRODUCTION

Service oriented approach has enabled development of new business processes as well as expansion of existing processes with value added services. For composition of such processes an organization does not need to rely on its own resources for software coding and computing infrastructure for realization and execution of the underlying business process tasks (Moser et al., 2008; Baresi and Guinea, 2011; Paliwal et al., 2012; Chun et al., 2005). Rather, the individual tasks of the business process are performed by invoking Web services offered by third parties. Especially in the e-commerce and financial domains, there are numerous third party Web services covering every major functional aspect such as invoicing and billing, taxation and costing, accounting, payment processing, shipment, and so on (Turban et al., 2009). The increased availability of such Web services has opened new opportunities for organizations for rapid and cost-effective development of their business processes even if they lack the capability to perform all of the tasks required for the business processes.

As an example, consider a business process for Internet orders processing by an online store. This is a complex process involving several tasks and each task itself can be a multi-step workflow as shown in Figure 1. However, all these tasks can be executed by utilizing existing third party Web services. Assuming that the business process workflow is known and all the relevant Web services have been selected a priori then the process can be invoked and executed anytime. However, coming up with the implementation level business process workflow design so that each task can be mapped to an individual Web service as well as selection of third party web services for such tasks, poses significant challenges (Paliwal et al., 2012). Typically, business process development involves working with a high-level process specification and manually elaborating the high-level specifications into implementation level design (Dengler et al., 2011; Chun et al., 2002b). After this, partner Web services are selected and a binding is established between the workflow tasks and the selected partner Web services to generate the executable business process (e.g., BPEL process (Evdemon et al., 2007)).

Such business process development can be done collaboratively by exploiting the knowledge of the existing business processes of related organizations to

112 Irshad H., Shafiq B., Vaidya J., Ahmed Bashir M., Shamail S. and Adam N..
Preserving Privacy in Collaborative Business Process Composition.
DOI: 10.5220/0005567801120123
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 112-123
ISBN: 978-989-758-117-5
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

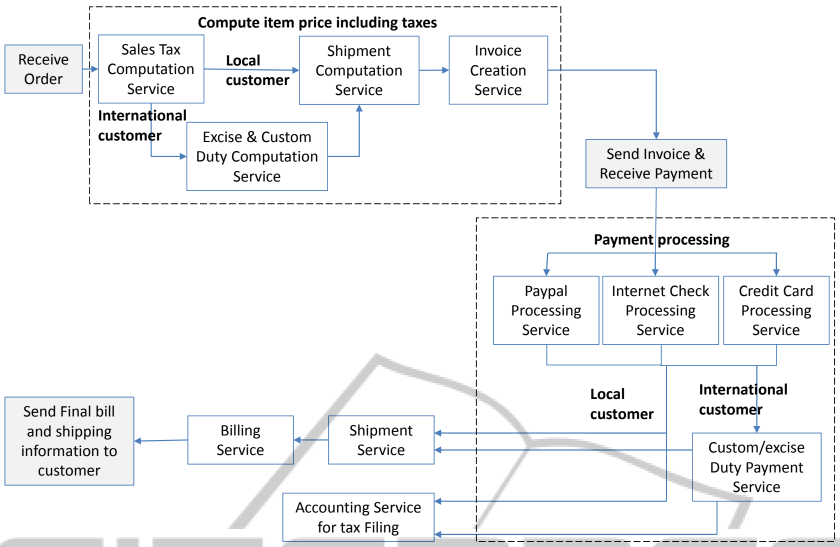

Compute item price including taxes
Receive Order
Sales Tax Computation Service
Local customer
Shipment Computation Service
Invoice Creation Service
International customer
Excise & Custom Duty Computation Service
Send Invoice & Receive Payment
Payment processing
Paypal Processing Service
Internet Check Processing Service
Credit Card Processing Service
Local customer
International customer
Send Final bill and shipping information to customer
Billing Service
Shipment Service
Custom/excise Duty Payment Service
Accounting Service for tax Filing

Figure 1: Example of a business process for handling Internet purchase orders.

compose an executable business process of a given organization based on its requirements and design specifications. However, this requires a repository of existing business processes that includes different types of business processes such as supply chain management, Internet purchase orders, accounting and taxation. For each type, the repository has to include the history of the business process invocations by different organizations in form of execution sequences (e.g., order in which the activities were performed and the corresponding Web services invoked). Given such a repository, we can learn the common execution patterns for different types of processes and use the learned patterns for composing a given type of business process for an organization (Bentounsi et al., 2012a; Bentounsi et al., 2012b).

For illustration, suppose that a garment store wants to develop a business process for handling Internet purchase orders. The store owner specifies the following requirements in the process design specification:

1) Orders need to be handled for both local and international customers.

2) For international orders the excise and custom duty must be charged from the customer in the payment amount and paid to the appropriate authority.

3) Each transaction needs to be recorded in the accounting system for filing of the sales and income tax returns.

Based on the given requirements, the process repository is searched to retrieve the set of execution sequences that process either local orders or International orders with customs and excise duty payment. This set of execution sequences can be used to determine: i) relevant activities in the Internet purchase order business process; ii) controlflow and dataflow between the activities; iii) third party web services that can perform the different activities in the business process; and iv) trustworthiness of such third party Web services based on their invocation frequency. Given this information, an executable business process can be composed and deployed. For example, we can employ process mining on the sequence dataset to regenerate the BP workflow (van der Aalst et al., 2010).

However, creating such a repository typically requires sharing sensitive information. Essentially, organizations with existing business processes, have to share and upload their process execution sequences to the central repository. The execution sequences (even after masking of the data values such as credit card information, social security number, etc.) include sensitive business information which organizations may not want to share (Kerschbaum and Deitos, 2008; Bentounsi et al., 2012a; Bentounsi et al., 2012b). For instance an organization may not want its competitors to learn: how many times the organization selects a given third party Web service as opposed to another service with the same functionality? How many times a business process failed for the organization because the requested item was not in the inventory? What is the turn-around time from order receipt to shipment?

To address this problem, we develop a privacy-preserving Business Process Recommendation and Composition System (BPRCS), that generates a differentially private dataset of execution sequences which can be published and shared with other organizations for composition and implementation of their business processes. We employ process mining and classification on the differentially private sequence dataset to regenerate the BP workflow. From this BP workflow, we can generate the executable BP code in BPEL.

As such, this paper examines the problem of how privacy can be preserved if an executable business process has to be composed based on the knowledge of existing business processes. This is an important problem since there is a lot of benefit in creating business processes that utilize the available web services. We do this by employing the notion of differential privacy to protect the confidentiality of information contributed by each organization. Our key contribution is to exploit the sequential composition property of differential private computations by modeling the execution sequences as a graph and performing random walk on this graph to regenerate comparable execution sequences that cannot be linked to a contributing organization.

The rest of the paper is structured as follows. The collaborative environment for business processes composition is discussed in Section 2. We present the privacy model in Section 3 and describe the proposed approach in Section 4. In Section 5, we analyze the privacy and computation complexity of the proposed approach. We present the results of the experimental evaluation in Section 6. In Section 7, we discuss related work and conclude the paper in Section 8.

2 ENVIRONMENT

Figure 2 shows the environment for business process recommendation and composition. BPRCS is a trusted party in the cloud that provides support for collaborative composition of executable business processes in a privacy-preserving manner. Organizations from different domains register with BPRCS and publish their business process details (BP execution sequences) which are stored in a repository after anonymization using the proposed differential privacy-based approach. The dataset in this repository can be made publicly available as the anonymization ensures that no business secret and private information can be learned from the anonymized BP execution sequences. BPRCS also maintains a registry of third party Web services and based on the information derived from the BP execution sequences in the repository it computes trustworthiness and other quality of service values for each Web service.

The users subscribe to BPRCS for using its services to query the repository for retrieving relevant business process sequences. The user may perform process mining on the retrieved sequences to learn the relevant activity patterns and compose the activity patterns into a business process based on the given requirements. Alternatively, the user may provide his/her requirements and high level process design to BPRCS which also includes a business process mining component for composition of a business process in an interactive manner. Inputs to the business processing mining component include the BP execution sequences from the repository as well as the available web services as shown in Figure 2. Using these inputs, the business process mining component generates the business process workflow with mapping of the activities to the available Web services. The user can modify the generated business process workflow using the process refinement interface. The process refinements may include changing the activity to Web service mapping or modifying the structure of the generated BP workflow. Given the business process workflow design and the activity to Web service mapping, an executable BPEL process can be easily generated and deployed on the server either at the user site or on the cloud.

One question that might occur is why would organizations participate in the BPRCS system. Basically, given an assurance of privacy, there are several incentives for these organizations to share their data in this collaborative environment, as discussed below:

1. Organizations can use the BPRCS for composing new business processes.
2. Refinement/extension of existing processes – An organization can compare its business process with comparable processes from other organizations to check for inconsistency/redundancy or identify any extension that may add more value to the process.
3. The proposed BPRCS system is also suitable for less open collaborative environments (e.g., digital government, single large organization) that require a pre-established level of trust before using third party Web services or ownership of such services (Chun et al., 2002a). In such environments, the organizations may be trustworthy; however, they may have similar privacy concerns for sharing their sensitive business process data. Also in such environments (e.g., digital government), there is typically a large number of trusted or co-owned Web services that could be used in composition of new business processes by other organizations. For example a property tax assessment Web service provided by County treasurer department can be used by the business registration department of the city. In this case, both the subscriber and user organizations are trustworthy

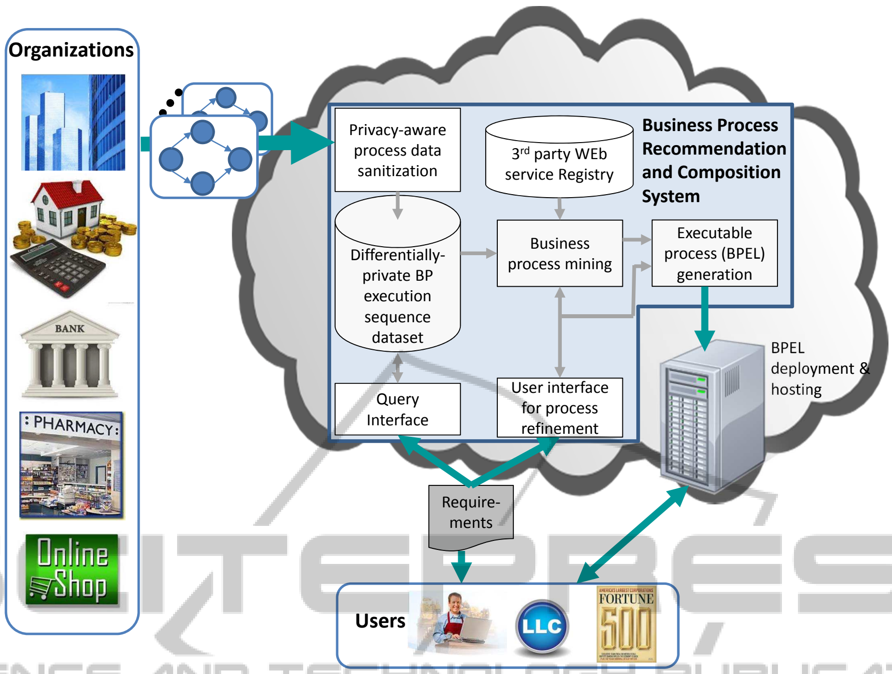

Organizations
PHARMACY
Online Shop
Privacy-aware process data sanitization
3rd party WEb service Registry
Business Process Recommendation and Composition System
Differentially-private BP execution sequence dataset
Business process mining
Executable process (BPEL) generation
BPEL deployment & hosting
Query Interface
User interface for process refinement
Require-ments
Users
LLC
FORTUNE 500

Figure 2: Collaborative environment for business process composition.

and use BPRCS for composition of new business processes.

3 PRIVACY MODEL

We first formally define business process execution sequences and then present the privacy model.

3.1 BP Execution Sequence

Generally, a business process is not designed for one time use only. Users execute their business processes repeatedly and each execution of the business process may be different from a prior execution of the same process. We define an execution sequence of a business process as the ordered list of activities (along with their input/output parameters) that are executed in any given execution of the business process. The activities in an execution sequence are ordered based on their invocation time. Based on BPEL formalism(Evdemon et al., 2007), an activity can be an invoke activity (invoking a Web service), receive activity (receiving input from a user), reply activity (sending a reply to the user), and assignment activity (for variable/value assignment). Since receive, reply, and assignment activities are primarily used to capture the data flow between Web services (invoke activities), we do not consider these activities in an execution sequence and define an execution sequence with respect to the Web services invoked. The data flow is captured by annotating each Web service with its input and output parameters in the given execution sequence. We represent the Web service invocation by the tuple

$$WS = (WS_{id}, input\_List, output\_List)$$

where WS_{id} corresponds to the unique identifier of the Web service being invoked. $input\_list = \{(input\_parameter, value)\}$ and $output\_List = \{(output\_parameter,\ value)\}$. Note that a given web service may provide multiple operations and in the BP execution sequence we need to capture which specific operation of the web service was invoked. However, for simplicity of discussion we assume that each Web service provides only one operation. In case a Web service provides multiple operations, we assign a unique WS_{id} to each combination of the Web service and its operation.

Definition 1 (BP Execution Sequence (ES))**.** *A BP execution sequence is a list of WS tuples ordered by their invocation time in the given business process execution.*

Table 1 shows a sample of BP execution sequences database including four sequences related to the Internet purchase order of Figure 1. The Web service invocation tuples in the sequence are separated by $\rightarrow$. The sensitive information in the input list and output list is masked by the organization sharing its sequence data. Only those input and output parameter values are disclosed which are not considered sensitive. For example, in the **GetOrderFromAmazon** Web service in sequence 1 (Table 1), all the input parameter values including customer name and address are masked. Similarly, all the input and output parameter values in the Web service tuple **EasyBill** are masked to protect leakage of sensitive information.

3.2 Differential Privacy

Differential Privacy (Dwork, 2006; Dwork et al., 2006) is a well accepted privacy model that provides a formal and quantifiable privacy guarantee irrespective of an adversary's background knowledge and available computational power. A randomized algorithm is considered to be differentially private if for any pair of neighboring inputs, the probability of generating the same output, is within a small multiple of each other, for the entire output space (Dwork, 2006). Thus, for any two datasets which are close to one another, a differentially private algorithm will behave approximately the same on both data sets.

Definition 2 (ε-Differential Privacy)**.** *A randomization algorithm $\mathcal{A}$ satisfies ε-differential privacy if for any two neighboring datasets D_1 and D_2 (differing in one element), and all $R \subseteq Range(\mathcal{A})$, we have* $e^{-\varepsilon} \leq \frac{Pr[\mathcal{A}(D_1) \in R]}{Pr[\mathcal{A}(D_2) \in R]} \leq e^{\varepsilon}$.

A standard mechanism to achieve differential privacy is based on the addition of appropriately parameterized Laplacian noise. For this, we need to define the sensitivity of the function to be computed.

Definition 3 (Sensitivity)**.** *For any query function q over the input datasets, the sensitivity of q is $\Delta q = max||q(D_1) - q(D_2)||$ for any neighboring datasets D_1 and D_2.*

Queries with lower sensitivity can better tolerate the data modifications from added noise. The work in (Dwork et al., 2006) shows that to release a (perturbed) value $q(x)$ while satisfying privacy, it suffices to add Laplace noise with standard deviation $\Delta q/\varepsilon$. More specifically, for any given query function q, the mechanism $\mathcal{A}$:

$$\mathcal{A}(D) = q(D) + Laplace(\Delta q/\varepsilon)$$

gives ε-differential privacy. Note that the above privacy guarantee requires that all the tuples in the dataset are independent (Kifer and Machanavajjhala, 2011), and may not hold if this is not true.

3.3 Privacy Requirement

Given a BP execution sequence database D and any BP execution sequence $ES \in D$, we do not want an adversary to learn if ES is the execution sequence of an organization Org_x. Thus, in terms of differential privacy, any two execution sequence databases D_1 and D_2 differing only on the inclusion of ES (i.e., D_1 includes ES and D_2 does not include ES), an ε-differentially private algorithm $\mathcal{A}$ satisfies:

$Pr[\mathcal{A}(D_1) \in R] = e^{\varepsilon} Pr[\mathcal{A}(D_2) \in R]$

Where $R \subseteq Range(\mathcal{A})$. As stated in Section 3.2, this privacy guarantee is based on the assumption that all the execution sequences in the datasets D_1 and D_2 are independent. This assumption would definitely be valid if each organization contributes at most one execution sequence to the database. In case an organization (Org_x) contribute m sequences to the execution sequence database, then the adversary's probability estimate that a given execution sequence belongs to Org_x can change by at most $e^{m\varepsilon}$ (Kifer and Machanavajjhala, 2011). Note that this will only be true if the execution sequences are all based on the same underlying business process. In any case, to avoid even the potential of such disclosure, we restrict the number of execution sequences contributed by each organization to 3.

4 PROPOSED APPROACH

We first discuss how the differentially privacy repository of BP execution sequences is generated and then describe process mining for BP workflow creation.

4.1 Differentially Private BP Execution Sequences

We model the BP execution sequence database as a graph $G = (V, E)$. A node in the graph represents a Web service and an edge represents ordering relationship between invocation of two web services in some execution sequence. We formally define the execution sequence graph below:

Definition 4 (BP Execution Sequence Graph)**.** *A BP execution sequence graph $G = (V, E)$ is a compact representation of the BP execution sequence database.*

- *Each node in V corresponds to a Web service.*
- *An edge $(WS_i, WS_j) \in E$ denotes that in some business process execution, the Web service WS_i was invoked first followed by the Web service WS_j.*
- *$c : E \to \mathbb{Z}^+$ is a count function. $c(WS_i, WS_j)$ denotes how many times WS_j was invoked after invocation of WS_i.*
- *$dist_{ip} : E \to \mathbf{X}$. For an edge $e : (WS_i, WS_j)$, $\mathbf{X}$ is a vector of distributions of all the input parameter*

Table 1: Sample BP execution sequence database.

No.	BP Execution Sequence
1	(**GetOrderFromAmazon**,{ }, {(Item,LevisJeans), (Qty,1) (BuyerLoc,LA)}) → (**SalesTaxSVC**$_A$,{(Item,LevisJeans), (Qty,1), (Price,50) (BuyerLoc,LA)},{(Tax,4)}) → (**TransShip**,{(wt,0.5), (Origin,NYC), (Dest,LA) (Type,Std)},{(Charge,5)}) → (**PayPal**, {(Amount,59)},{(Result,OK) }) → (**EasyBill**, { },{ })
2	(**GetOrderFromAmazon**,{ }, {(Item,LevisJeans), (Qty,1) (BuyerLoc,London)}) → (**SalesTaxSVC**$_A$,{(Item,LevisJeans), (Qty,1), (Price,50) (BuyerLoc,London)},{(Tax,4)}) → (**UKCustoms**,{(ItemType,Clothes), (Qty,1), (Price,54),(BuyerLoc,London},{(Duty,2)})) → (**TransShip**,{(wt,0.5), (Origin,NYC), (Dest,LA) (Type,Std)},{(Charge,5)}) → (**PayPal**, {(Amount,61)},{(Result,OK) }) → (**UKDutyPymt**, {(Amount,2)},{(Result,OK) }) (**EasyBill**, { },{ })
3	(**GetOrderFromEbay**,{ }, {(Item,PoloShirt), (Qty,1) (BuyerLoc,Chicago)}) (**EasySalesTax**,{(Item,PoloShirt), (Qty,1), (Price,60) (BuyerLoc,Chicago)},{(Tax,5)}) → (**USPS**,{(wt,0.5), (Origin,NYC), (Dest,Chicago) (Type,Urgent)},{(Charge,10)}) → (**PayPal**, {(Amount,75)},{(Result,OK) }) → (**EasyBill**, { },{ })
4	(**GetOrderFromAmazon**,{ }, {(Item,LevisJeans), (Qty,2) (BuyerLoc,Paris)}) → (**SalesTaxSVC**$_A$,{(Item,LevisJeans), (Qty,2), (Price,100) (BuyerLoc,Paris)},{(Tax,8)}) → (**EuroCustoms**,{(ItemType,Clothes), (Qty,1), (Price,54),(BuyerLoc,Paris)},{(Duty,2)}) → (**TransShip**,{(wt,1), (Origin,NYC), (Dest,LA) (Type,Std)},{(Charge,10)}) → (**PayPal**, {(Amount,120)},{(Result,OK) }) → (**EuroDutyPymt**, {(Amount,2)},{(Result,OK) }) (**EasyBill**, { },{ })

values of WS_j given that WS_j is invoked immediately after WS_i. If WS_j has n input parameters then **X** *includes n distributions.*

- $dist_{op} : E \rightarrow \mathbf{Y}$. *For an edge* $e : (WS_i, WS_j)$, **Y** *is a vector of distributions of all the output parameter values of WS_i given that WS_j is invoked immediately after WS_i. If WS_i has n output parameters then* **Y** *includes n distributions.*
- $slen : V \rightarrow 2^{\mathbb{Z}^+ \times \mathbb{Z}^+}$. *For a node v,* $slen(v) = \{(length, count)\}$. $slen(v)$ *returns the length of all execution sequences that originate from v. count denotes the number of occurrences of the sequence.*

In the execution sequence graph, the functions $dist_{ip}$ and $dist_{op}$ keep track of the distribution of the input and output parameter values of the Web services against each edge. This distribution of parameter values would be needed in the process mining phase to identify any conditional branches when composing the business process.

Figure 3 shows the graph of based representation of the sample BP execution sequence database of Table 1. This graph has 11 nodes corresponding to each Web service in the sample database. The edges in the graph are labeled with the count value given by the function $c()$. For example, $c(v_1, v_3) = 3$ indicating that there are three sequences in which the Web service **SalesTaxSVC**$_A$ was invoked after the **GetOrderFromAmazon** Web service. The input parameters value distribution for the edge (v_1, v_2) is also shown in Figure 3. The distribution indicates that amongst the corresponding sequences, the service **SalesTaxSVC**$_A$ was invoked 3 times when the item was *LevisJeans*

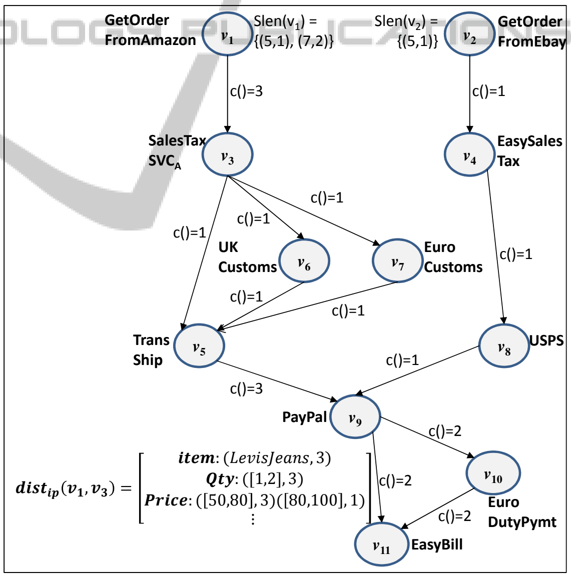


Figure 3: Graph based representation of the sample BP execution sequence database of Table 1.

and quantity was in the range [1-2]. Also, in these three invocations the price was in the range [50-80] twice and in the range [80-100] once. As discussed in Section 3.2, the values of sensitive parameters in the execution sequences would be masked by the organizations for privacy concerns. Therefore, the value distributions of those parameters are not included in $dist_{ip}$ and $dist_{op}$. The starting nodes in the sample database are v_1 corresponding to **GetOrderFromAmazon** and v_2 corresponding to **GetOrderFromEbay**. $slen(v_1) = (5,1), (7,2)$ indicating that there is one sequence of length 5 and two sequences of length

Algorithm 1: Generate Differentially Private BP Execution Sequences.

```
Require: BP execution sequence database $D = \{ES_1, ES_2, \ldots, ES_N\}$
Require: Privacy measure $\varepsilon$
Ensure: $\varepsilon$-Differentially private BP execution sequence database $\tilde{D}$
 1: $\tilde{D} \leftarrow \phi$
 2: From the given database $D$, generate the execution sequence graph $G = (V, E)$
 3: For each edge $e \in E$, label $e$ with $c(e)$, $dist_{ip}(e)$, and $dist_{op}(e)$.
 4: For each node $v \in V$, label $v$ with $slen(v)$
 5: $k \leftarrow max(\{length(ES_1), \ldots length(ES_N)\})$
 6: for each $e \in E$ do
 7:     $c(e) \leftarrow c(e) + Laplace(\varepsilon/k)$ {Add Laplacian noise to the edge count}
 8: end for
 9: Based on the modified count value for an edge $e = (WS_i, WS_j)$, compute the probability of taking that edge, i.e., the probability of invoking service $WS_j$ immediately after invocation of $WS_i$
10: for each $v \in V$ do
11:     for each $t \in slen(v)$ do
12:         for $i = 1$ to $t.count$ do
13:             Generate $\widetilde{ES}$ by doing a random walk of $t.length$ steps starting from node $v$ using the probability values of corresponding edges computed in line 9
14:             $\tilde{D} \leftarrow \tilde{D} \cup \widetilde{ES}$ (this is a multiset union)
15:         end for
16:     end for
17: end for
18: return $\tilde{D}$
```

7 that originate from v_1. For all nodes other than v_1 and v_2 $slen()$ is an empty set.

We use the BP execution sequence graph to generate the differentially private database of execution sequences. The basic idea is to add Laplacian noise with appropriate ε value to the count value of each edge. Based on the edge count values, we compute the probability of taking an edge from a given node. For example in Figure 3 there are three edges originating from node v_3: (v_3, v_5), (v_3, v_6), and (v_3, v_7). Each has a count value of 1, so the probability of each edge is $1/3$. Given the edge traversal probabilities, we can do a random walk from the starting nodes to regenerate the BP execution sequences. The length and the number of such execution sequences from any given starting node v is given by the $slen(v)$

Algorithm 1 shows the pseudo code for generation of differentially private BP execution sequence database. The input to this algorithm is the original sequence database and the privacy budget ε. The output database generated by the algorithm ensures ε-differential privacy. Algorithm 1 first generates the BP execution sequence graph from the given database (lines 1-4). Then noise derived from Laplace distribution with 0-mean and $\lambda = \varepsilon/k$ is added to the count value for each edge (lines 6-8). k is the length of the longest sequence in the database D. The steps for regeneration of BP execution sequences by doing random walk from the start nodes are shown in lines 10-17.

4.2 Process Mining and BP Workflow Generation

Once the differentially private BP sequence database is generated, it can be queried to retrieve relevant sequences based on user requirements as depicted in Figure 2. From the retrieved BP sequences, we need to generate the BP workflow structure including the control flow and data flow. In addition the activities in the BP workflow need to be mapped to appropriate Web services.

There are several process mining tools available for discovering workflow models from event logs (van der Aalst et al., 2010; Van der Aalst et al., Sept; Silva et al., 2005; Wen et al., 2009). We use the process mining tool ProM (van der Aalst et al., 2010) to create this basic workflow structure from the given BP execution sequences. From this basic workflow structure, we identify all the branching points. There may exist a strong correlation between the values of some of the input/output parameters and the branch taken during business process execution. For example, with reference to the Internet purchase order example depicted in Figure 1, if the order is made by an international customer, the business process branches to the excise and custom duty computation service. On the other hand, if the order is made by a local customer the business process execution takes the first branch. For each branching point, we build a decision tree-based classification model to discover the branching conditions. The differentially private execution dataset (used for generation of the basic workflow) serves as a training dataset for classification models generation. Essentially, each BP execution sequence in this dataset corresponds to an instance with relevant data and parameter values for building the classification model. The rules generated by the classification model corresponds to the branching conditions and are annotated on the corresponding branching path.

Note that given the workflow structure including the controlflow, dataflow, activities to Web services mapping and branching condition, we can create the executable BPEL process that can then be deployed.

5 ANALYSIS

Privacy Analysis. As discussed in Section 3.3, the differential privacy guarantee requires that there must not be any correlation between multiple sequences submitted by the same organization that enables one to link such sequences to that organization. In case an organization (Org_x) contribute m sequences to the execution sequence database, then the adversary's probability estimate that a given execution sequence belongs to Org_x can change by a factor of $e^{m\varepsilon}$ (Kifer and Machanavajjhala, 2011).

In our experiments, we limit the maximum number of execution sequences contributed by any organization to 3 (i.e., $m \leq 3$). Also, when considering multiple BP execution sequences from a single organization, we ensure that such sequences do not overlap or the overlap is frequent across multiple organizations such that the information about the overlapping sequences does not significantly increases the probability of linking the overlapping sequences to specific organizations.

Below, we discuss the privacy analysis with respect to generation of differentially private BP execution sequence database (Algorithm 1). The process mining step operates on this differentially-private database and therefore does not affect privacy.

To satisfy the differential privacy requirement, we add Laplace noise to the edge count values in Algorithm 1. The standard deviation λ of Laplace noise depends on two parameters *sensitivity* and the privacy budget ε.

Sensitivity Δq. The count value for an edge (v_1, v_2) basically gives the number of BP execution sequences in which Web service v_2 was invoked immediately after v_1. Given that the neighboring database for differential privacy differ in one record, the sensitivity value for count queries is 1 (McSherry, 2009).

Privacy budget. Algorithm 1 regenerates the BP execution sequences by doing random walk up to some given length. In the random walk, basically we are composing the sequence of computations that each provide differential privacy in isolation. Here, the computations are the count queries on the edges. Assuming that a computation C_i provides ε_i differential privacy, then by the sequential composition theorem in (McSherry, 2009), the sequence of computations $C_i(D)$ provides $(\Sigma_i \varepsilon_i)$-differential privacy. To ensure $m\varepsilon$ differential privacy for the released database $\tilde{D}$, we consider a privacy budget of $m\varepsilon/k$ when adding noise to count value to edges, where k is the maximum length of any execution sequence in the original database D and m is the maximum number of execution sequences contributed by any organization. Since, we can have at most k steps in the random walk for creating any execution sequence and $\Sigma_{i=1}^{k}(m\varepsilon/k) = m\varepsilon$, therefore our algorithm provides at least $m\varepsilon$-differential privacy.

Computation Complexity. We first discuss the complexity of Algorithm 1. Lines (1-4) of Algorithm 1 generate the execution sequence graph $G = (V, E)$. The total number of nodes ($|V|$) in G is equal to the total number of web services in the BP sequence Database D and the number of edges can be at most $|V|^2$. Therefore, the size of G is $O(|D|)$. In lines (6-8) Laplace noise is added to each edge, therefore, the complexity of this step is $O(|E|)$. In the BP execution sequence regeneration step, we do a random walk as many times as the number of sequences in D. In each random walk, at most k nodes are visited, where k is the length of the longest sequence in D. Therefore, the runtime complexity of the sequence regeneration step is $O(k|D|)$. Hence, the overall complexity of Algorithm 1 is $O(k|D| + |E|) = O(k|D|)$.

The computation complexity for workflow generation from the BP execution sequences depends on the complexity of the process mining and classification model generation. We use ProM tool that incorporates α-algorithm for process mining (Van der Aalst et al., Sept). The complexity of α-algorithm is linear in the number of sequences and exponential in the number of tasks/activities in the workflow (Van der Aalst et al., Sept). The number of tasks/activities in a workflow does not depend on the size of the BP sequence dataset and is typically less than 100. Therefore, complexity is not a major issue for process mining. The classification model is built at each branching point to identify the branching conditions. The computation complexity of building the classifier depends on the particular classification model used. For example the cost of building an ID3 decision tree classifier is $O(|D||A||N|)$, where, $|D|$ is the number of sequences, $|A|$ is the number of attributes, and $|N|$ is the number of non-leaf nodes in the decision tree built (Quinlan, 1986).

6 EXPERIMENTAL EVALUATION

In this section, we evaluate the utility of the proposed approach in terms of the semantic correctness of the BP workflow generated from the differentially-private dataset. We refer to the BP workflow generated from the differentially-private dataset as output BP workflow. We measure the semantic correctness of the output BP workflow with respect to the following requirements:

1. **Completion of the Output BP Workflow**. This requirement entails that the output BP workflow terminates in the correct state. We measure the utility with respect to this requirement by computing the degree of overlap between the set of terminating states of the output BP workflow and original BPs (from which the BP sequence dataset is generated).
2. **Dependency Preservation in the Output BP Workflow**. Ensuring that the output BP workflow terminates in the correct state is not sufficient to verify its correctness. We also need to make sure that in any possible execution of the output BP workflow, the ordering and dependence relationship between the different activities are maintained with respect to the original BPs. Since there is a large number of original BPs (over 1000), we cannot perform structural comparison between the set of original BPs and the output BP workflow. Rather, we compare the overlap between the frequent sequential patterns of originals BPs and the frequent sequential patterns in the sanitized (differentially private) sequence dataset.

6.1 Dataset Description

For experimental evaluation, the dataset includes execution sequences selected from BPs related to Internet purchase orders. Since our experiments involved over a thousand BPs, we could not get these many real-world BPs. To address this issue, we first identified a large set of Web services and their categories related to Internet purchase orders in different businesses (e.g., garments/apparel, electronics). Then we identified the ordering and dependence relationships (e.g., shipment Web service is called after payment processing web service) and mutual exclusion constraints (e.g., Payment can be made using either Paypal or credit card Web service but not both) between the different Web service categories. These relationships and constraints were modeled in a dependency graph in which the nodes represent Web service categories. We consider two types of edges: i) dependence edge defining the dependence relationship between the Web service categories; and ii) constraint edges defining the mutual exclusion constraint between the Web service categories. From this dependency graph, we generated the required number of BPs by considering different combinations of alternate paths to terminal nodes as well as considering different parallel structures for independent nodes

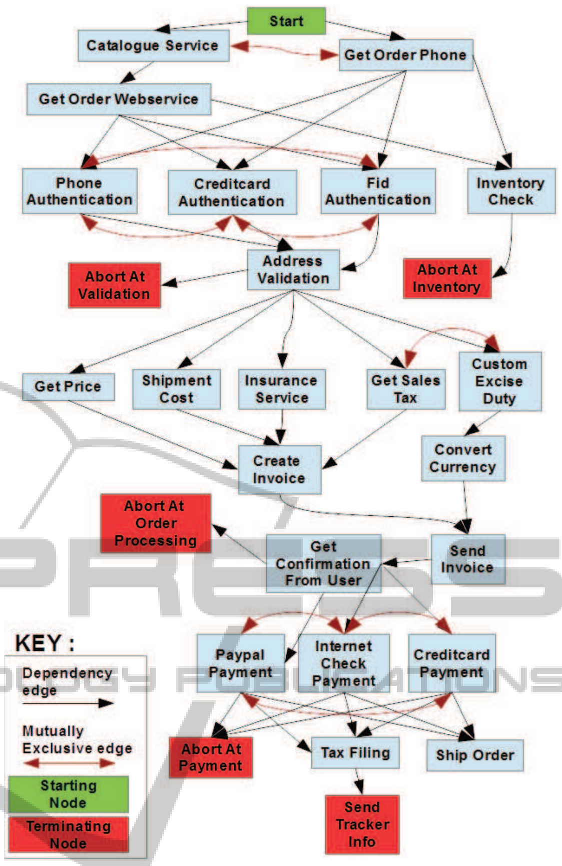


Figure 4: Dependency graph of Internet Purchase Order process.

(i.e., Web services that do not have any dependence relationship). Figure 4 depicts the dependency graph of the Internet purchase order.

6.2 Results

In the following discussion, D denotes the database of the original execution sequences generated from all the BPs of a dependency graph. D_S denotes the database of the BP sequences used for sanitization $D_S \subseteq D$. Each BP contributes a small number (1 – 3) of execution sequences in D_S for privacy reasons discussed above. $\tilde{D}$ denotes the sanitized database after applying Algorithm 1 on D_S.

6.2.1 Completion of the Output BP Workflow

For this requirement, we measure the degree of overlap between the set of terminating states of the output BP workflow and original BPs (from which the BP sequence dataset is generated). Specifically, we measure the accuracy of the set of terminating states in the original BP workflow (denoted by T) and the set of terminating states (denoted by $\tilde{T}$) in the output

Table 2: Precision and recall of terminating states with varying $\tilde{\varepsilon}$ and number of BPs related to Internet purchase order dataset.

	Precision (Percent)				**Recall (Percent)**			
$\tilde{\varepsilon}$	200 BPs	500 BPs	1000 BPs	2000 BPs	200 BPs	500 BPs	1000 BPs	2000 BPs
0.02	100	100	100	100	60	60	60	60
0.01	100	100	100	100	40	60	60	60
0.005	100	100	100	100	40	40	60	60
0.002	100	100	100	100	40	60	60	60

BP workflow in terms of precision and recall. These values were averaged over four runs. $\tilde{T}$ is obtained from the output BP workflow structure returned after performing process mining on the sanitized execution sequence dataset.

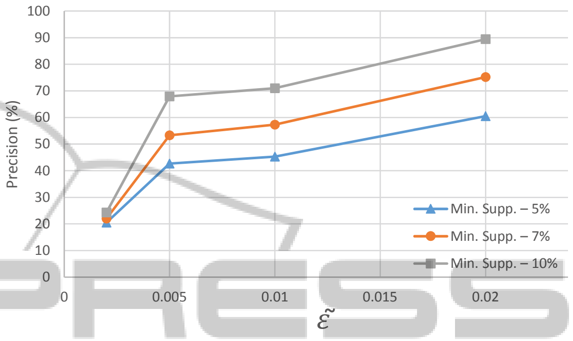
Table 2 shows precision and recall values of terminating states with varying $\tilde{\varepsilon}$ (Note that $\tilde{\varepsilon} = \varepsilon/k$ and for the Internet Purchase Order BPs $k \leq 16$. Thus $\tilde{\varepsilon}$ represents the privacy budget for reconstructing each step of the business process execution sequence.) For this experiment, we considered different numbers of BPs (200, 500, 1000, and 2000). These BPs were generated from the Internet Purchase Order dependency graph. From each BP, we selected 3 execution sequences and sanitized the resulting database of selected execution sequences.

As shown in Table 2, Precision is 100% for the Internet purchase order dataset. Recall improves as number of BPs are increased for generation of the output BP workflow. Also, recall improves as we decrease the noise (i.e., increase $\tilde{\varepsilon}$)

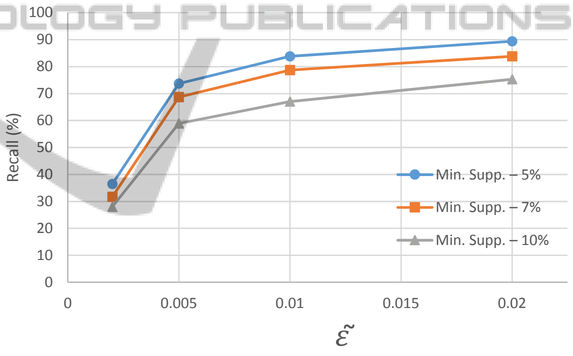
6.2.2 Dependency Preservation in the Output BP Workflow

We compute the overlap between the frequent sequential patterns in D and $\tilde{D}$ in terms of precision and recall. For measuring recall, we considered exact matching of patterns between D and $\tilde{D}$. For measuring precision, we consider two patterns p_i and p_j to be matching if they differ by at most one element and the order of elements is preserved. This approximate matching (for precision) makes sense since business process composition is an interactive process. The user will take the output of the system and refine it based on his/her requirements.

Precision and Recall of Frequent Sequential Patterns w.r.t. ε. Figure 5 (a) and (b) shows the graph of precision and recall values measured against different $\tilde{\varepsilon}$ for the Internet Purchase Order BPs. The precision and recall values in Figure 5 are computed by taking the average of precision and recall values of the frequent patterns with length between 5 and 10 in Figure 5, the number of input BPs is 1000. From each of the input BP, we randomly selected 1 execution sequence for generation of the output BP workflow. We



(a) $\tilde{\varepsilon}$ vs. precision

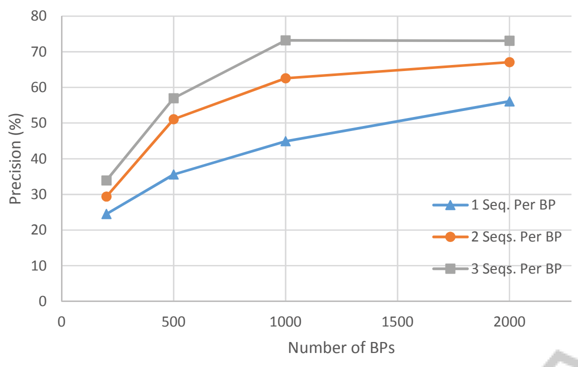


(b) $\tilde{\varepsilon}$ vs. recall

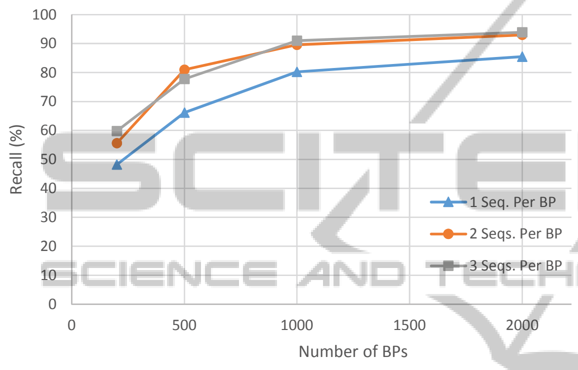
Figure 5: No. of BPs = 1000 and 1 sequence per BP for Internet Purchase Order dataset.

first ran the frequent sequential pattern analysis on the original database D with a minimum support threshold of 10% for all patterns of length between 5 and 10. For the sanitized database $\tilde{D}$, we considered 3 minimum support thresholds: 5%, 7%, and 10% in Figure 5. The reason for choosing smaller support threshold (5% and 7%) for differentially private dataset is that addition of noise typically decreases the support value for frequent patterns. However at 5% minimum support and $\tilde{\varepsilon} \geq 0.005$ more than 75% of all the frequent patterns of the original dataset are retrieved as depicted in Figure 5(b). As depicted in Figure 5, both precision and recall increases as the value of $\tilde{\varepsilon}$ increases (i.e., lesser the noise, better the results.)

Precision and Recall of Frequent Sequential Patterns w.r.t. Number of BPs. Figure 6 (a) and (b) shows the graph of average precision and recall values measured against different number of input BPs for



(a) No. of BPs vs. precision



(b) No. of BPs vs. recall

Figure 6: $\tilde{\varepsilon} = 0.01$ and minimum support threshold for frequent patterns in D = 10% and for $\tilde{D}$ = 5% – Internet Purchase Order dataset.

the Internet Purchase Order. In this graph, $\tilde{\varepsilon} = 0.01$, minimum support threshold for frequent patterns in D is set to 10% and for $\tilde{D}$ is set to 5%. From each of the input BP, we randomly selected 1, 2, and 3 execution sequence for generation of the output BP workflow. As depicted in Figure 6, the precision and recall improves as the number of input BPs increases. Moreover, the results improve with the increase in the number of execution sequences from each BP for generation of the output BP workflow.

7 RELATED WORK

Differential Privacy for Sequence Dataset. (Chen et al., 2012) have proposed a differential privacy based approach for sanitization of trajectory data. In this approach they first build a prefix tree from the given trajectory dataset and then add Laplacian noise at each level of the prefix tree to satisfy differential privacy. From the noisy prefix tree, they regenerate trajectories for the sanitized dataset. Their approach can be an alternate approach for generation of the sanitized BP Execution sequence database as trajectories are essentially sequence of locations. However, in their prefix-tree based approach, there can be multiple branches in the prefix tree for sequences that overlap considerably but differ only in the beginning part of the sequences. Due to this, the memory requirement for generation of the prefix tree for BP execution sequence dataset will be high.

Collaborative Business Process Composition. There is some work on collaborative design and composition of business processes by utilizing social network platforms (Koschmider et al., 2010; Dengler et al., 2011; Bruno et al., 2011; Brambilla et al., 2012). These social network platforms serve as a recommendation system that facilitates a process designer to complete or update a formal BP model on the basis of the prior usage of the process fragments by other peers. However, the privacy issues related to sharing of the usage patterns and process fragments are not considered.

(Bentounsi et al., 2012b; Bentounsi et al., 2012a) have proposed an anonymization-based approach for privacy preserving outsourcing of business processes in a multi-tenant cloud environment. In their approach a business process is broken down into smaller fragments for re-use in future process modeling. A fragment is disclosed for process composition only if it satisfies the K_l *anonyfrag* requirement. This requirement enatils that the disclosed fragment must have at least K clones distributed among l tenants. This *anonyfrag* approach indeed satsifies the privacy requirement. However, it requires the process designer to manually compose the business process from the available fragments. In contrast, our proposed approach generates the initial BP workflow (based on the given requirements) that can be refined by the process designer.

8 CONCLUSION

In this paper, we have described a privacy preserving approach to enable collaborative composition of executable business processes. Organizational business process details in form of BP execution sequences are collated together in a differentially private central repository that supports process mining for generating business processes based on users requirements and high level design specifications. The results from the experimental evaluation show that the proposed approach is effective in terms of preserving utility of the original BP execution sequence database while guaranteeing privacy. This utility is measured with respect to preservation of process workflow structure and semantics as well as preservation of Web service ranking.

One limitation of the proposed approach is that the central repository is trusted to sanitize the data before making it public. We can employ encryption based techniques to address this limitation which would add significant computation and communication overhead. In the future, we plan to develop efficient techniques to deal with cases in which the central repository cannot be fully trusted.

ACKNOWLEDGEMENTS

The work of Irshad and Shafiq is supported by the LUMS Faculty Initiative Fund Grant. The work of Vaidya is supported by the National Science Foundation under Grant No. CNS-1422501.

REFERENCES

Baresi, L. and Guinea, S. (2011). Self-supervising bpel processes. *IEEE Trans. Softw. Eng.*, 37(2):247–263.

Bentounsi, M., Benbernou, S., and Atallah, M. J. (2012a). Privacy-preserving business process outsourcing. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 662–663.

Bentounsi, M., Benbernou, S., Deme, C. S., and Atallah, M. J. (2012b). Anonyfrag: An anonymization-based approach for privacy-preserving bpaas. In *Proceedings of the 1st International Workshop on Cloud Intelligence*, Cloud-I '12, pages 9:1–9:8.

Brambilla, M., Fraternali, P., and Vaca, C. (2012). Bpmn and design patterns for engineering social bpm solutions. In *Business Process Management Workshops*, pages 219–230.

Bruno, G., Dengler, F., Jennings, B., Khalaf, R., Nurcan, S., Prilla, M., Sarini, M., Schmidt, R., and Silva, R. (2011). Key challenges for enabling agile bpm with social software. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):297–326.

Chen, R., Fung, B. C., Desai, B. C., and Sossou, N. M. (2012). Differentially private transit data publication: a case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 213–221.

Chun, S., Atluri, V., and Adam, N. R. (2002a). Dynamic composition of workflows for customized egovernment service delivery. In *Proceedings of the 2002 annual national conference on Digital government research*, pages 1–7.

Chun, S. A., Atluri, V., and Adam, N. R. (2002b). Domain knowledge-based automatic workflow generation. In *Database and Expert Systems Applications*, pages 81–93.

Chun, S. A., Atluri, V., and Adam, N. R. (2005). Using semantics for policy-based web service composition. *Distributed and Parallel Databases*, 18(1):37–64.

Dengler, F., Koschmider, A., Oberweis, A., and Zhang, H. (2011). Social software for coordination of collaborative process activities. In Muehlen, M. and Su, J., editors, *Business Process Management Workshops*, volume 66 of *Lecture Notes in Business Information Processing*, pages 396–407.

Dwork, C. (2006). Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I., editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third conference on Theory of Cryptography*, TCC'06, pages 265–284.

Evdemon, J., Arkin, A., Barreto, A., Curbera, B., Goland, F., G.Kartha, Khalaf, L., Marin, K., van der Rijn, M., and Yiu, Y. (2007). Services business process execution language version 2.0. *OASIS Standard*.

Kerschbaum, F. and Deitos, R. J. (2008). Security against the business partner. In *Proceedings of the 2008 ACM workshop on Secure web services*, SWS '08, pages 1–10.

Kifer, D. and Machanavajjhala, A. (2011). No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204.

Koschmider, A., Song, M., and Reijers, H. A. (2010). Social software for business process modeling. *Journal of Information Technology*, 25(3):308–322.

McSherry, F. D. (2009). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 19–30.

Moser, O., Rosenberg, F., and Dustdar, S. (2008). Non-intrusive monitoring and service adaptation for ws-bpel. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 815–824.

Paliwal, A. V., Shafiq, B., Vaidya, J., Xiong, H., and Adam, N. (2012). Semantics-based automated service discovery. *Services Computing, IEEE Transactions on*, 5(2):260–275.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Silva, R., Zhang, J., and Shanahan, J. G. (2005). Probabilistic workflow mining. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 275–284.

Turban, E., Lee, J. K., King, D., Liang, T. P., and Turban, D. (2009). *Electronic commerce 2010*.

Van der Aalst, W., Weijters, T., and Maruster, L. (Sept.). Workflow mining: discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1128–1142.

van der Aalst, W. M., Pesic, M., and Song, M. (2010). Beyond process mining: from the past to present and future. In *Advanced Information Systems Engineering*, pages 38–52.

Wen, L., Wang, J., Aalst, W., Huang, B., and Sun, J. (2009). A novel approach for process mining based on event types. *Journal of Intelligent Information Systems*, 32:163–190.