DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
38

GAMAL ELKOUMY

Privacy-Enhancing Technologies for Business Process Mining

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

38

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
38

GAMAL ELKOUMY

Privacy-Enhancing Technologies for Business Process Mining

UNIVERSITY OF TARTU
Press
1632

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on 2nd November, 2022 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof.	Marlon Dumas University of Tartu, Estonia

Opponents

Prof.	Han van der Aa University of Mannheim, Germany
Assist. Prof.	Marwan Hassani Eindhoven University of Technology, The Netherlands

The public defense will take place on December 12, 2022, at 10:15 a.m. at Delta Building, Narva mnt 18, Room 1021.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906 (print) ISSN 2806-2345 (PDF)
ISBN 978-9916-27-074-5 (print) ISBN 978-9916-27-075-2 (PDF)

University of Tartu Press
`http://www.tyk.ee/`

ABSTRACT

Process mining techniques enable organizations to analyze business process execution traces to identify opportunities for improving operational performance. The applicability of process mining techniques hinges on the availability of event logs capturing the execution of a business process. These event logs may contain private information in some use cases, particularly those involving customer-facing processes. Data protection regulations, such as the GDPR, restrict the use of such event logs for analysis purposes. In such cases, organizations need to deploy Privacy-Enhancing Technologies (PETs) to strike a balance between the benefits they get from analyzing these data and the requirements imposed on them by privacy regulations. In particular, this thesis aims to minimize re-identification risks (e.g., singling out of individuals) when event logs are disclosed to a process analyst within an organization and to enable the secure inter-organizational process analysis. The main contributions of this thesis are four approaches that focus on applying PETs to process mining to help organizations to achieve privacy-preserving process mining. Each technique tackles a specific setting to provide the user with various alternatives that best suit the situation at hand. We present event log anonymization approaches that provide proven differential privacy guarantees. Also, we offer an approach that adopts the secure multi-party computation protocol to enable multiple organizations to analyze their shared logs jointly. We conduct an empirical evaluation showing that the proposed approaches outperform the state-of-the-art in terms of data utility (accuracy) loss and computational efficiency. Furthermore, we provide the end user with GDPR-compliant open source tools to mitigate their privacy risks.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Business Process Intelligence Challenge	BPIC
Business Process Model and Notation	BPMN
Contingency Table	CT
Cumulative Distribution Function	CDF
Deterministic Acyclic Finite State Automata	DAFSA
Differential Privacy	DP
Directly-Follows Graph	DFG
Earth Movers' Distance	EMD
Garbled Circuit	GC
Homomorphic Encryption	HE
Hospital Information System	HIS
Jaccard Distance	JD
Multi-Party Computation	MPC
Predicate Singling Out	PSO
Prior Knowledge	PK
Privacy-Enhancing Technologies	PETs
Privacy-Preserving Process Mining	PPPM
Renyi Differential Privacy	RDP
Single-Instruction Multiple-Data	SIMD
Social Security Number	SSN
Structured Query Language	SQL
Symmetric Mean Absolute Percentage Error	SMAPE
eXtensible Event Stream	XES

1. INTRODUCTION

In order to fulfill the needs of their customers and stakeholders, organizations need to continuously optimize their day-to-day operations. One way of achieving this continuous optimization imperative is by putting in place initiatives to understand, analyze, redesign and monitor their *business processes*. A business process is "a collection of inter-related events, activities, and decision points that involve a number of actors and objects, which collectively lead to an outcome that is of value to at least one customer" [1].

To analyze business processes and discover different process execution flows, organizations use a tool called *process mining*. Process mining is "a family of techniques to analyze the performance and conformance of business processes based on event logs produced during their execution" [1]. Process mining techniques allow organizations to dig deeper and understand the reasons behind certain bottlenecks. Furthermore, process mining enables organizations to check the applicability of their business rules inside their environment and quantify and visualize their performance.

The input of process mining techniques is an event log. An event log captures the execution of a set of process instances (herein called *cases*). Table 1 shows an example of an event log from a healthcare system. An event log consists of event records. Each record contains a reference to a case identifier, a reference to an activity, and at least one timestamp. Each case ID refers to a person (e.g., a patient in a hospital). Each event corresponds to an activity performed by a resource for that person or an interaction with that person. For example, in a healthcare process, each activity may correspond to a treatment the patient in question underwent, and a resource may correspond to the doctor performing the activity. The sequence of activities that have been executed for an individual is known as *case variant*. For instance, the case variant of patient 1 is *Register*, *Visit*, *Blood Test*, and *Discharge*.

Table 1. A fragment of a simplified event log obtained from a hospital: each row corresponds to an event.

ID	Activity	Timestamp	Resource	Role	Age	Sex	Zip	Disease
1	Register	07.01.2020-08:30	ResA	Admin	22	M	13053	Flu
1	Visit	07.01.2020-08:45	ResE	Doctor	22	M	13053	Flu
2	Register	07.01.2020-08:46	ResA	Admin	30	F	13061	Infection
3	Register	07.01.2020-08:50	ResA	Admin	32	F	51009	Infection
1	Blood Test	07.01.2020-08:57	ResB	Admin	22	M	13053	Flu
1	Discharge	07.01.2020-08:58	ResC	Admin	22	M	13053	Flu
2	Hospitalize	07.01.2020-09:01	ResD	Admin	30	F	13061	Infection
3	Hospitalize	07.01.2020-10:00	ResD	Admin	32	F	51009	Infection
2	Blood Test	07.01.2020-10:02	ResB	Nurse	30	F	13061	Infection
3	Blood Test	07.01.2020-10:15	ResB	Nurse	32	F	51009	Infection
2	Blood Test	07.02.2020-08:00	ResB	Nurse	30	F	13061	Infection
2	Visit	07.02.2020-09:30	ResE	Doctor	30	F	13061	Infection
3	Visit	07.02.2020-13:55	ResE	Doctor	32	F	51009	Infection
2	Discharge	07.02.2020-14:00	ResF	Admin	30	F	13061	Infection
3	Discharge	07.02.2020-14:15	ResF	Admin	32	F	51009	Infection

Event logs may contain private information in some use cases, particularly those involving customer-facing processes. Let us consider the situation where an analyst happens to know that John underwent a Blood test for a Flu diagnosis. Given this information and the released Table 1, the analyst can link the first case to the patient John, and they can disclose the information about John in the log. In other words, the analyst can *single out* John.

Data minimization principles embedded in privacy regulations, such as GDPR [2], require that organizations put in place mechanisms to protect information about individuals when processing a dataset. The definition of individual data identification is articulated in Recital 26 of GDPR:

> To determine whether a natural person is identifiable, account should be taken of all the means reasonably likely to be used, such as singling out, either by the controller or another person, to identify the natural person directly or indirectly.

The notion of singling out is elaborated in a guide by the Article 29 Data Protection Working Party [3]. According to this guide, a person can be identified (singled out) using a dataset if the dataset allows us to distinguish that person from all other persons represented in the dataset. In other words, a dataset allows a singling out of an individual (from a group) if a predicate can be evaluated on that dataset and uniquely distinguishes the individual in question. The legal notion of singling out has been formalized by Cohen & Nissim [4]. Specifically, Cohen & Nissim give a mathematical formulation of the concept of Predicate Singling Out (PSO), capturing the idea that there exists a predicate that uniquely identifies a row in a dataset.

The applicability of process mining techniques hinges on the availability of event logs capturing the execution of a business process. To be able to analyze privacy-sensitive event logs within the framework of GDPR and similar privacy regulations, organizations need to make use of Privacy-Enhancing Technologies (PETs) [5].

A PET is a type of technology that allows to either release a dataset or a query answer, or to perform computations on one or more datasets, while controlling the ability for an analyst and/or an adversary to view or infer personal information about individuals represented in the dataset(s).

1.1. Problem Statement

Among existing PETs, Differential Privacy(DP) stands out because it mitigates PSO attacks [4] and because it provides composable privacy guarantees [6]. In a nutshell, differential privacy guarantees to each individual that the ability for the analyst to infer private information about them will likely be the same whether their record(s) is/are part of the dataset or not. In this way, DP ensures that the analyst will not be able to single out an individual based on the anonymized data disclosed to them [4].

DP works by injecting noise into the data. This noise is quantified by a *privacy budget* parameter called ε. The value of ε captures the extent to which the presence or absence of the record(s) associated with a person in the original data affect the disclosed (anonymized) data. The smaller the ε value, the stronger the privacy guarantee and the larger the injected noise. Thus, by controlling the privacy budget, we can ensure that the resulting anonymized dataset provides privacy guarantees while still being useful for analysis. For example, in a healthcare event log, we can control the ε privacy budget to ensure that we can identify bottlenecks and improvement opportunities from the anonymized log while providing privacy guarantees to the patients whose clinical trajectories are represented by the traces in the log. In other words, the privacy budget can be seen as a *privacy risk threshold* that controls the level of provided privacy guarantee.

In this setting, this thesis addresses the following research question:

RQ1 *Given an event log L, wherein each trace contains private information about an individual (e.g., a customer), and given a privacy risk threshold, how to generate an anonymized event log L' that provides a differential privacy guarantee to each individual represented by a trace in the log, w.r.t. the given threshold?*

Sometimes, the ε parameter of differential privacy may not be interpretable to end users without a deep understanding of differential privacy. A user may find it hard to determine a suitable value of ε. In this respect, Lee et al. [7] show that "the proper value of ε varies depending on individual values" and that the presence of "outliers also changes the appropriate value of ε." Meanwhile, Dwork et al. [8] state that "we do not know what parameter ε is right for any given differentially private analysis and we know that the answer can vary tremendously based on attributes of the dataset." Accordingly, some privacy mechanisms determine the ε value required to anonymize an event log based on a business-level privacy risk threshold, namely *guessing advantage*. The guessing advantage is the increase in the probability that an adversary may guess information about an individual after the event log's release. This thesis presents approaches that address **RQ1** with respect to both the privacy risk thresholds: ε and guessing advantage.

With respect to **RQ1**, we should ensure that the anonymized log is useful for process mining. In other words, a process mining algorithm should return a similar result with the anonymized log as it would with the original one. To this end, a desirable property is that the anonymized log should have the same set of case variants as the original one. A case variant is a distinct sequence of activities. This property ensures, for example, that the set of directly-follows relations between activities is not altered during anonymization. This set of relations, known as the Directly-Follows Graph (DFG), is used by automated process discovery techniques [9]. Moreover, the set of case variants is the main input used by conformance checking techniques [10]. In this setting, preserving the set of case variants during anonymization is critical, as every case variant added to the log directly is

a potential false positive (a deviation that does not exist in reality but is reported as such). In contrast, every removed case variant is a potential false negative (a deviation occurring in reality but not detected when using the anonymized log). In such settings, PETs for process mining that address **RQ1** should fulfill the following utility requirement:

UR1a *The anonymized event log should have the same set of case variants as the original log.*

When an event log contains a case variant that occurs only once in the log (unique traces), the above requirement may turn out to be overly strict and may require the differential privacy mechanism to inject a high level of noise. Indeed, to hide the presence of an individual represented by a unique trace, the differential privacy mechanism may need to insert many other synthetic traces. In some settings, the user may accept suppressing some traces in order to reduce the noise level required to achieve a given privacy guarantee. In other words, the user may settle for a relaxed version of requirement **UR1a**, such that some case variants may be suppressed, but no new case variants are introduced. This remark leads to the following alternative utility requirement:

UR1b *The anonymized event log must not introduce new case variants to the original log.*

A third desirable property is that the differences between the timestamps of consecutive events in the anonymized log are as close as possible to those in the original log, as these time differences are used by performance mining techniques [11]. Therefore, we should address **RQ1** while keeping the differences between the anonymized log and the original log minimal. The difference between the anonymized and original logs is called *utility loss*. Accordingly, we introduce the following additional utility requirement:

UR2 *The difference between the real and the anonymized time values is minimal given a privacy risk metric.*

The requirement **UR2** can be tackled w.r.t. different attack models. This thesis considers an attack model wherein the attacker seeks to single out an individual, represented by a trace in the log, based on a prefix, a suffix, or an event timestamp of the individual's trace in the released log.

The anonymization techniques studied in this thesis focus on protecting the personal information of the subjects (e.g., patients) whose customer journeys are represented in the event log. In the event log of Table 1, the focus of this thesis is on anonymizing logs to protect the patients (John, Anna, Lena, etc.). The thesis is not concerned with the orthogonal problem of protecting the personal information of the Resources (workers) in the process, i.e., the personal information of the people references in the third column of the event log in Table 1.

Besides, even if we put aside the above challenges of anonymizing data for process mining within a single organization. Existing process mining techniques require access to the entire event log of a business process. Usually, this require-

ment can be fulfilled when the event log is collected from one or multiple systems within the same organization. In practice, though, many business processes involve multiple independent organizations. We call such processes *interorganizational business processes*. This collection step should precede the anonymization step to make the combination possible. Exchanging execution data may reveal the personal information of customers, or it may expose business secrets. As a result, common techniques for process mining cannot be employed for interorganizational business processes. Yet, analyzing these processes is often crucial for improving operational performance. Given the above limitation of applying PETs for process mining, this thesis addresses the following research question:

RQ2 *How to enable process mining for interorganizational business processes without requiring the involved parties to share their private event logs or trust a third party?*

1.2. Previous Work and Research Gaps

The use of privacy-enhancing technologies for process mining has been considered in a body of studies known as *privacy-preserving process mining* (PPPM). The problem stated in RQ1 has been addressed partially using k-anonymization approaches. These approaches suppress entire cases or individual events (activity instances). When a technique suppresses an activity instance in a case, it may introduce behavior (e.g., a directly-follows relation) that does not exist in reality. However, the k-anonymity is not secure against predicate singling out attacks [4]. Indeed, Cohen et al. [4] proved that under some settings, the above approaches enable an attacker to perform a predicate single-out attack with a probability of 37%. When this suppression occurs at scale, the anonymized log contains a high proportion of behavior that does not happen in the original log. In a patient treatment log used in the evaluation reported in this paper, existing k-anonymization techniques for process mining may lead to the suppression of 87% of the activity instances [12]. The current k-anonymity approaches does not fulfill UR1a.

Other approaches address RQ1 via differential privacy. These approaches inject noise into the data, quantified by a privacy budget parameter called ε. Differentially private approaches for event log anonymization inject noise to anonymize the frequency distribution of distinct sequences of activities in a log (the trace variants) and the event timestamps. This noise injection may introduce behavior not observed in the original log. The extent to which a DP-anonymization technique distorts the data and increases the utility loss. The holy grail of anonymization techniques in general, and DP-anonymization techniques in particular, is to achieve a low level of re-identification risk with low utility loss. The current differential privacy approaches do not fulfill the utility requirements UR1a, UR1b and UR2.

1.3. Research Method

Since our research area is in the field of information systems, to answer the above research questions, we follow the design science guidelines in information system research proposed by Hevner et al. [13]. They proposed the continuous construction and evaluation of design artifacts to address a research question and improve the performance of artifacts. Therefore, we identified the methodological limitations of the current privacy-preserving process mining techniques, defined the research questions (RQ1 and RQ2), and deduced open research gaps in the current state-of-the-art to answer these questions.

Accordingly, we defined a set of requirements for constructing three artifacts. The first artifact to address RQ1 and fulfill requirement UR1 via the oversampling as a method to inject noise into the event log. While constructing the first artifact, we conducted a benchmark that allowed us to relax the requirement UR1a to UR1b and adopt the idea of personalized differential privacy and a filtration technique to reduce the utility loss of the anonymized log.

In the second artifact, we leverage the idea of privacy amplification via subsampling to address RQ1 and achieve a more utility-friendly anonymization approach. In the third artifact, we addressed RQ2 using the multi-party computation protocol to build process maps in an interorganizational setting.

Furthermore, Hevner et al. [13] emphasizes the need to validate the proposed artifacts in real or realistic settings to ensure that they fulfill the stated requirements. To evaluate the proposed artifacts, we use real-life experiments, with respect to evaluation questions derived from the statement of the research questions and the associated requirements.

1.4. Contributions

This thesis provides the below four contributions to the field of privacy-preserving process mining:

Contribution 1: proposes a differentially-private mechanism to release event logs while maintaining the same set of trace variant as the original log (Chapter 4). To this end, we propose a method to generate a differentially private event log by oversampling the traces of the original log and injecting noise to the event timestamps. The proposed method is parameterized by a user-defined maximum risk level, δ. The method ensures that, after the differentially private event log is released, the probability that an attacker may single out an individual, based on the prefixes and suffixes of the traces in the log, and based on the event timestamp, does not increase by more than δ, relative to the baseline scenario where the differentially private event log is not released.

The proposal relies on a data structure that compactly captures all prefixes and suffixes of a set of traces, namely a Deterministic Acyclic Finite State Automata (DAFSA) [14]. A DAFSA is a lossless representation of an event log, wherein

every prefix or suffix shared by multiple traces is represented once. By analyzing the frequency and time differences of each DAFSA transition, we determine the amount of oversampling and timestamp noise.

Contribution 2 proposes a differentially-private mechanism to release event logs without introducing new case variants to the anonymized log (Chapter 5) To this end, we apply both over and undersampling of traces that exist in the event log. We filter out high-risk cases to lower the utility loss. We used personalized differential privacy to quantify the required noise amount for each event in the log. Also, we compare the proposed approach against state-of-the-art and assess utility loss not only w.r.t. the distance between the anonymized and the original log, but also w.r.t. the impact of anonymization on the process maps discovered from the (anonymized) log.

Contribution 3 proposes an approach that enhances the noise injection of the differentially-private mechanism of event logs using privacy amplification (Chapter 6). We leverage the idea that the privacy guarantees of a differentially private mechanism can be amplified by applying it to a small, random subsample of records. We hypothesize that a DP approach based on subsampling can achieve lower utility loss for a given level of privacy guarantee relative to existing DP-anonymization techniques for event logs, which are based purely on noise injection. We start by filtering out trace variants that, if disclosed, would lead to privacy breaches. We then extract multiple Poisson subsamples and apply a DP mechanism to anonymize each subsample. The resulting differentially private subsamples are then combined to construct an anonymized log. Using the differential privacy composition theorem and the privacy amplification results associated with *Renyi Differential Privacy* (RDP), we estimate the amplified ε' privacy guarantee provided by the resulting anonymized log.

Contribution 4 proposes an approach to enable process mining in an interorganizational setting (Chapter 7). To this end, we propose an architecture for process mining based on secure multi-party computation. In essence, MPC aims at the realization of some computation over data from multiple parties while exposing only the result of the computation but keeping the input data private. We consider the setting of an MPC platform, where the involved parties upload their event logs to a network of compute nodes. Before the upload, secret sharing algorithms locally split each data value into different parts (i.e., shares) that are then stored at different nodes. Each share does not provide any information about the original data. The uploaded event log is encrypted and exposed neither to the platform operator nor to other involved parties. Nonetheless, the MPC platform enables the computation of the encrypted data through protocols to hand out the result among the nodes.

1.5. Outline

The rest of this thesis is structured as follows. In Chapter 2, we introduce background notions and concepts of privacy-enhancing technologies. In Chapter 3, we identify different threats of publishing event logs and present design requirements from the GDPR for PPPM. Also, we discuss existing techniques and the room for improvements. Chapter 4 introduces the notion of a differentially-private event log and proposes oversampling as a way to inject noise. Chapter 5 introduces the notion of personalized differential privacy and proposes both over-and undersampling to hide the presence and absence of individuals. Chapter 6 proposes and evaluates a method that reduces the noise injection of differentially-private event logs. Chapter 7 proposes and evaluates the usage of a secure multi-party computation protocol to enable PPPM in an interorganizational setting. Chapter 8 concludes this thesis and discusses future research directions. Table 2 summarizes the research questions addressed, their associated utility requirement, the contributions that address each research gap, and the chapter of the thesis where this contribution is presented.

Table 2. Research questions, requirements, thesis contributions and publications.

Research Questions	Requirements	Contributions
RQ1: Given an event log, and given a maximum acceptable risk threshold, how to generate an anonymized log such that the risk of singling out an individual after releasing the anonymized log does not exceed the given threshold, and the difference between the real and the anonymized time values is minimal?	UR1a and UR2.	**1.** Mine Me but Don't Single Me Out: Differentially Private Event Logs for Process Mining. (See Chapter 4) **Publication:** • (2021) ICPM21 Conference paper [15]
	UR1b and UR2	**2.** Differentially Private Release of Event Logs for Process Mining. (See Chapter 5) **Publications:** • (2022) ICPM22 Demo paper
	UR1b	**3.** High-Utility Anonymization of Event Logs for Process Mining: A Subsampling Approach. (See Chapter 6) **Publication:** • (2022) ICPM22 Conference paper [16]
RQ2: How to enable process mining for interorganizational business processes without requiring the involved parties to share their private event logs or trust a third party?	-	**4.** Secure Multi-Party Computation for Inter-Organizational Process Mining. (See Chapter 7) **Publications:** • (2021) BPMDS20 Conference paper [17] • (2020) BPM20 Demo paper [18]

2. BACKGROUND

The applicability of process mining techniques hinges on the availability of event logs capturing the execution of a business process. These event logs may contain private information in some use cases, particularly customer-facing processes. With automatic process discovery, we can reverse engineer the activities that have been performed within the organization and the hand-offs between resources that execute such activities. Hence, it is easy to expose confidential information with process mining techniques.

Data protection regulations, such as the GDPR, restrict the use of such event logs for analysis purposes. Hence, privacy is becoming a legal responsibility rather than a social one. Accordingly, it is necessary to incorporate privacy aspects in the discovery, analysis, and monitoring phases. One way of circumventing these restrictions is to use privacy-enhancing technologies to give access to process mining approaches to private event logs. The main objective of this chapter is to introduce the concepts and techniques of privacy-enhancing technologies that could be adapted for process mining.

This chapter is structured as follows. Sect. 2.1 presents the definition of privacy and confidentiality. Sect. 2.2 introduces process mining and the typical components of a process mining system. Then, we introduce different privacy techniques as follows, in Sect. 2.3, we define the group-based privacy techniques, in Sect. 2.4, we introduce differential privacy guarantees and formalize its definitions, and in Sect. 2.5, we give an overview of secure multi-party computation protocols.

2.1. Privacy and Confidentiality

Privacy and confidentiality have a lot in common that may lead to confusion; however, each has a specific meaning.

Privacy. In our current data-driven society, privacy has received much attention through frequent data breaches and regulations such as Europe's General Data Protection Regulation (GDPR) [2]. Generally, privacy is seen as the right of individuals to control how their personal data is collected, used, and/or disclosed to other individuals, organizations, or governments [19]. GDPR defines personal data as "Personal data means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person can be identified, directly or indirectly [..]" [2]. Besides GDPR, privacy is subject to other international laws such as the UN Declaration of Human Rights [20], and the Asia-Pacific Economic Cooperation [21]. We follow the definitions of GDPR in this paper.

Confidentiality. Whereas there is some overlap and the concepts are often used interchangeably, the focus of confidentiality is ensuring that only authorized individuals have access to the protected data and information. For example, Harman et al [22] defined as an agreement about maintenance and who has access to classified/sensitive data. Thus, confidentiality concerns data access, while privacy

focuses on individuals and their rights. When the data are *personal data*, the confidentiality challenges coincide with the privacy challenges. Here, we distinguish privacy from confidentiality as follows. When the main concern is an individual's rights, e.g., process workers, customers, or patients, it is considered a privacy issue. Otherwise, if the concern is more relevant to general data protection, it is assumed as a confidentiality issue.

2.2. Process Mining: Preliminaries

We introduce process mining and the typical components of a process mining system based on an example scenario in a hospital setting. Health care has seen many process mining applications [23, 24] which makes this a representative example. The goal of analyzing processes with process mining in our scenario is to prevent rework, decrease waiting time for patients, and improve documentation by discovering cases of non-compliance. The hospital is also interested in benchmarking, i.e., investigating how their processes and their performance differ from other hospitals.

Concretely, the hospital wants to apply process mining to discover the trajectory of different patients from the moment they are admitted until their discharge from the hospital. Each visit of a patient to the hospital forms a process instance or case, and the individual events of each case are sourced from the Hospital Information System (HIS). The HIS records information on logistical and treatment activities conducted for specific patients and who of the hospital staff performed them. In addition, a part of the process is performed via e-mails, e.g., referrals to other care institutions and the request of previous medical documentation. Therefore, certain events are collected from the e-mail server of the hospital. E-mails are associated with certain process activities using text mining and the metadata (sender, recipient) is used to identify the staff responsible. Finally, to benchmark, the hospital wants to share some of this data over organizational boundaries and perform interorganizational process mining [25, 26].

Based on this scenario, we describe the elements of such a typical process mining application. We organize the elements into three layers: data, application, and presentation layer.

Data Layer.. Process mining starts from event data, a collection of events or event log representing the execution of several instances of a business process. Table 1 shows an example of such an event log extracted from the HIS and e-mail system of the hospital. Here, each row represents an event that indicates when an activity was performed (*Timestamp*) and by whom it was performed (*Resource*). Furthermore, each event is associated with a running process instance or case (*Patient*), which in our case is a unique identifier of the patient visit. By grouping events based on their case and ordering them according to their timestamp, we obtain sequences of events: one *trace* for each process instance. In addition event logs often include additional domain specific attributes, which are not strictly

required for the application of basic process mining techniques, but may provide additional context. In our hospital scenario, these attributes are *Age*, *Sex*, *Zip* and *Disease*.

Definition 2.2.1 (Event Log, Event, Trace)**.** An event log $L = \{e_1, e_2, ..., e_n\}$ of a process is a set of events $e = (i, a, ts)$, each capturing an execution of an activity a (an activity instance), with a timestamp ts, as part of a case i of the process. The trace $t = \langle e_1, e_2, ..., e_m \rangle$ of a case i is the sequence of events in L with identifier i, ordered by timestamp. An event log L may be represented as a set of traces $\{t_1, t_2, ..., t_k\}$.

If we set aside the case ID and the timestamp of a trace and focus on the activity labels, we can represent a trace as a word over the alphabet of activity labels. Each particular word extracted from a log in this way is called a *case variant* of the log. For example, in Table 1, the case variant for *patient 1* is the sequence of activities: *Register*, *Visit*, *Blood Test*, and *Discharge*.

Definition 2.2.2 (Case Variant)**.** A trace variant of an event log L is a sequence of activities $\langle a_1, a_2, ..., a_k \rangle$ such that there is a trace $\langle e_1, e_2, ..., e_k \rangle$ of L such that $\forall j \in [1..k] : e_j = a_j$.

Application Layer. Algorithms that process event logs and compute representations are an important part of process mining. Two of the most important applications of process mining are *process discovery* [9] and *conformance checking* [27]. Process discovery receives an event log and returns a process model that describes the process behavior in an abstract model defining the possible sequences of activities. An example of such model that could be discovered from the log in Table 1 would be the BPMN[1] model shown in Figure 1. Here, the process discovery algorithm inferred from the event log that each trace starts with an activity *Register*. Then, activity *Hospitalize* must be performed in parallel with one or multiple *Blood Tests*. Finally, each case of our simplified process is concluded by a *Visit* in sequence with *Discharge*. Conformance checking aims to quantify deviations between the process model and real execution data as observed through the event log. The output of a conformance checking algorithm is usually information about how individual traces in the given event log deviate from that reference model. In Figure 1, a conformance checking technique has identified that the activity *Blood Test*, which occurred according to an event in the trace for patient 1, should not have been performed since the patient was not hospitalized. Many other tasks are possible such as the mining of resource profiles of the employees [28], the mining of decision rules [29] based on the characteristics of cases captured in additional attributes, or the automated prediction of the next step in a running case [30] with the goal of acting on cases leading to a bad process outcome or performance.

Presentation Layer.. Generated artifacts such as the discovered process models or the deviations detected by conformance checking, or other analytical outputs

[1]Business Process Model and Notation (BPMN) standard: https://www.omg.org/spec/BPMN/.

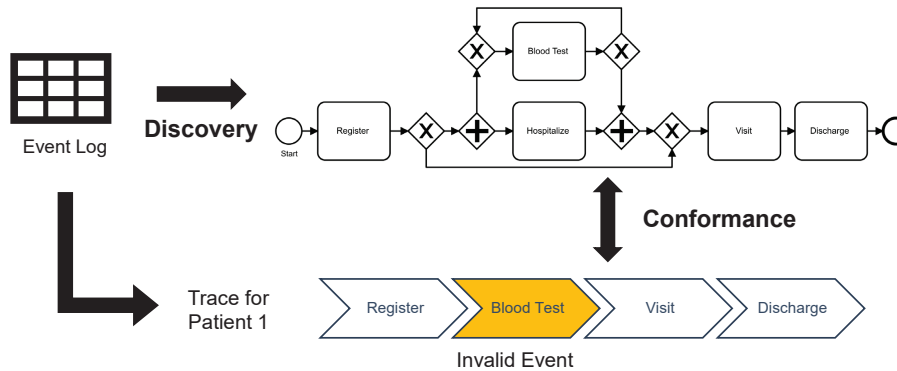


Figure 1. Process discovery and conformance checking shown based on the event log from Table 1. A process model in BPMN was discovered and used for conformance checking towards the first trace of the log. One of the events of the trace (*Blood Test*) should not have occurred according to the model.

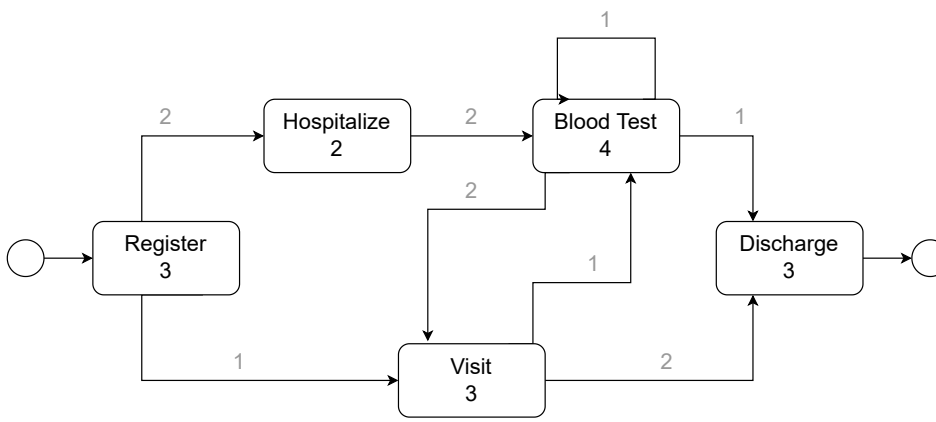


Figure 2. A frequency annotated DFG based on the combined results of process discovery and conformance checking. When only considering the presentation layer of the process mining application, the analyst may only see such a visualization of the underlying event log. Occurrence frequencies of activities (in black) and of the transitions between activities (in gray) are added as annotation to the model.

need to be presented to the process analyst or business user. Often these outputs are aggregated representations, e.g., *Directly-follows Graph (DFG)* (a.k.a. *process map*) with projected frequencies as shown in Figure 2. In addition to being a common approach to visualize the dependencies between activities in a process, the DFG is used as an intermediate artifact by various algorithms for automated discovery of process models [9].

A DFG is a directed graph in which each node represents activity in the process, and each arc represents a directly-follows relation between two activities, meaning that the activity that is the source of the arc was observed right before the activity

that is the target of the arc at least once in the event log. Typically, each arc in the DFG is annotated with the number of times that the target activity immediately follows the source activity (arc frequency). However, it may also be annotated with other metrics, such as the average timestamp difference between the source and the target activity or the maximum timestamp differences between the source and the target.

Definition 2.2.3 (Directly-Follows Graph)**.** The directly-follows graph of an event log L is $G(L) = \{(D_1, R_1), (D_2, R_2), ..., (D_n, R_n)\}$, with:

- $D = \{(x,y) | x \in A \wedge y \in A \wedge (x <_L y)\}$ is a pair of two activities and,
- R is the set of edges of the graph with weights $W = \{w | w \in \mathbb{R}\}$ and,
- $x <_L y$ iff $\exists t = \langle e_1, e_2, ..., e_m \rangle$and $i \in \{1, ..., m-1\}$ such that $t \in Traces(L)$ and $t_i.a = x$ and $t_{i+1}.a = y$

The weights W on the edges of the DFG are the result of an aggregate query function f, that represents the directly-follows relation between the pair of values D. An aggregation function f can be the count of occurrences as in the frequency annotated DFG, or an aggregation function over the time differences between the pair of values, e.g., the max time difference , as in the time annotated DFG.

We consider a scenario where an event log owner wants to enable an analyst to analyze their process execution without linking the record to the users whose information is recorded. Given the log in Table 1, we can give the following classification of privacy-related attributes:

- Case ID: This attribute is directly used to identify a case. It can link the case to a specific individual. For example, in Table 1, if the analyst knows that the case ID of John is 1, regardless of the existence of the name, they can link the first case to John.
- Case attributes: which are the attributes that remain with the same value for a case instance during the case execution, e.g., name, email, Social Security Number (SSN).
- Case variant: which is the sequence of activities that have been executed for the case. For example, in Table 1, given that Anna is the only patient with the case variant that contains two *Blood Tests*, the attacker can link case 2 to Anna.
- Quasi Identifiers: attributes that, if combined, can identify an individual (age, gender, profession, race, zip code). For example, in Table 1, given that *Lena* is the only 32 years old female patient from the zip code *51009*, the analyst can link case number 3 to *Lena*.
- Sensitive Attributes: personal attributes that should not be revealed (diagnosis, sexuality, salary, credits, payment transactions, financial situation). For example, in Table 1, the Diagnosis column is the sensitive attribute that we do not want the analyst to guess about the individuals.

At first glance, one can notice that the Case IDs and case attributes can single

out individuals in the event log. A naive way of hiding the identity of individuals is through *pseudonymization*. Pseudonymisation removes the identifying information like case ID, name, and email. This removal can also be achieved by hashing the values of these attributes. However, that would not be enough to hide the identity of individuals. For instance, if we removed both the case ID and the case attributes, we still know some details like gender, age, profession, race, and zip code. By any chance, if that person is the only female PhD student of age 32 from her village, then we can easily link the case to that person.

Furthermore, if we eliminated all the attributes except for the basic minimum representation of an event log which contains a pseudonymized case ID, the timestamp of executing every activity, and the activity label. In that case, an analyst can use the sequence of activity execution, i.e., trace variant, to link a case to an individual. For instance, if the analyst knows that the individual of interest has been through the reception desk at that specific time, or that the individual of interest is the only patient with breast cancer in this hospital. Then, it would be possible for the analyst to link the case to a certain individual. In the following sections, we explain different privacy-enhancing technologies that can be used for process mining. In section 3.1, we explain in details different threats on confidentiality and privacy of process mining.

2.3. Group-based privacy Techniques (M1)

Group-based privacy techniques require that an individual cannot be singled out within a group of similar individuals of a specific size. One approach that provides privacy guarantees over the identifying attributes is k-anonymity. An event log satisfies the k-anonymity requirements when an analyst who knows only the quasi-identifiers of the individual cannot single them out with a probability greater than $1/k$. k-anonymity is an easy-to-understand privacy model. However, it does not provide sufficient privacy guarantees against attribute disclosure [31].

Machanavajjhala et al. [32] introduce a stronger privacy notion called l-diversity. An event log is said to provide l-diversity guarantees when each sensitive attribute of each similar group of traces has at least l values. l-diversity privacy guarantee addresses the attribute disclosure. However, it is pruned against skewness and similarity attacks [31].

t-closeness is another group-based approach to protect attribute disclosure [31]. An event log is said to provide t-closeness guarantees if, for all groups of traces, the distance between the distribution of a disclosed attribute in the group and the distribution of the entire attribute in the event log is at most t.

While the above privacy models are interpretable and easy to understand, they are not secure against predicate singling out attacks [4]. Indeed, Cohen et al. [4] proved that under some settings, the above approaches enable an analyst to perform a predicate single out attack with a probability of 37%.

2.4. Differential Privacy (M2)

Differential privacy allows us to protect records collected from individuals in the context of a study or other data collection exercise. In this context, differential privacy is a guarantee to the individuals that the impact on them will be the same whether they were in the study or not. Differential privacy ensures that the same conclusions from the data will be reached. For example, in a healthcare system, an analyst can use the data to draw conclusions about the bottlenecks in the business process without identifying who are the individuals included in the analysis.

Specifically, differential privacy ensures that the revealed study results from the event log would "essentially" equally occur whether an individual participates in the study or not. Accordingly, the probabilities are drawn by random choices of the data owner, and the term essentially is controlled by a parameter called ε .

Differential privacy is a guarantee, not an algorithm. Therefore, for any data analysis task T, and a given value for ε, there will be many differentially private algorithms to compute T. The smaller ε gives stronger privacy guarantee, however, it is challenging to get accurate results when ε is small.

2.4.1. Models of Computation

In this thesis, we consider a setting where a trusted data owner holds an event log L that contains the personal data of individuals, and they are welling to give access to an analyst who draws conclusions from L. In that setting, we assume hat each individual's participation in L is captured by a single trace, and the privacy mechanism is to protect each individual's trace while permitting the data analysis tasks over L.

Usually, data owners can give access to their event logs in two ways: non-interactive, and interactive model. In the non-interactive model, data owners construct a sanitized version of their logs, or build a synthetic event log that contains summary statistics. After this one-shot release, the data owner has no control over the released event logs. In the interactive mode, the data analyst can send queries adaptively to the event log and gets sanitized responses. That gives flexibility to the data analyst to request new queries based on the observed responses. In this thesis, we use differential privacy in the context of constructing a sanitized (or anonymized) version of an event log.

2.4.2. Formalizing Differential Privacy

Now, we are ready to formally define differential privacy. As explained above, an event log L is a set of traces $\{t_1, t_2, ..., t_k\}$. A trace corresponds to an individual (e.g., a customer) who requires their privacy to be maintained. A mechanism $M : L \rightarrow Range(M)$ maps an event log L to a particular distribution of values $Range(M)$ (e.g., to a vector of real numbers). A privacy mechanism M can be either unbounded or bounded ε-differentially private (ε-DP). An unbounded ε-DP mechanism makes it hard to distinguish two event logs that differ in the *presence*

of one trace [33]. A bounded ε-DP mechanism makes it hard to distinguish two event logs that differ in the *attribute values* of one trace.

Definition 2.4.1 (Unbounded ε-differentially private mechanism [33])**.** A mechanism M is said to be ε-differentially private if, for all the event logs L_1 and L_2 differing at most on one trace, and all $S \subseteq Range(M)$, we have $Pr[M(L_1) \in S] \leq exp(\varepsilon) \times Pr[M(L_2) \in S]$.

Definition 2.4.2 (Bounded ε-differentially private mechanism [34])**.** A mechanism M is ε-differentially private if, for all the event logs L_1 and L_2 differing at most on the attribute values of one trace, and all $S \subseteq Range(M)$, we have $Pr[M(L_1) \in S] \leq exp(\varepsilon) \times Pr[M(L_2) \in S]$.

In some cases, it is desired to apply DP to only values of a particular attribute A, e.g., the attribute timestamp in Table 1. We apply DP to L_1 and L_2 w.r.t the attribute A, i.e., L_1 and L_2 differ only on A's value in a single trace. Moreover, we want to take into account the particular *amount of change* in attribute A.

Definition 2.4.3 (Bounded ε-differentially private mechanism w.r.t attribute)**.** A mechanism M is ε-differentially private w.r.t attribute A iff for every pair of event logs L_1 and L_2 differing along attribute A in at most one trace, and for all $S \subseteq Range(M)$, we have $Pr[M(L_1) \in S] \leq exp(\varepsilon \cdot |L_1.A - L_2.A|) \times Pr[M(L_2) \in S]$.

The ε-differential privacy restricts the ability to single out an individual (Def. 2.4.1 and 2.4.2) or disclose an individual's private attribute (Def. 2.4.3). In an interactive mechanism [35], a user submits a query function f to an event log and receives a noisified result. Formally, there is a mechanism M_f that computes f and injects noise into the result. The amount of noise depends on the *sensitivity* of f, which quantifies how much change in the input of f affects change in its output.

Definition 2.4.4 (Global Sensitivity)**.** Let $f : L \to \mathbb{R}^d$.

- Global sensitivity w.r.t. presence of a trace is $\Delta f = \max_{L_1,L_2} |f(L_1) - f(L_2)|$;
- Global sensitivity w.r.t. attribute A is $\Delta^A f = \max_{L_1,L_2} \frac{|f(L_1) - f(L_2)|}{|L_1.A - L_2.A|}$;

where max is computed over all event logs L_1, L_2 differing in one trace at most.

Given the event log L and the query function f, a randomized mechanism M_f returns a noisified output $f(L) + Y$, where Y is a noise value drawn randomly from a particular distribution. E.g., we can draw values from a Laplace distribution $Lap(\lambda, \mu)$, which has a probability density function $\frac{1}{2\lambda} exp(-\frac{|x-\mu|}{\lambda})$, where λ is a scale factor, and μ is the mean. It is known [6] that, for real-valued f, if we set $\mu = 0$, for $\lambda = \frac{\Delta f}{\varepsilon}$ we obtain an ε-DP mechanism w.r.t. a trace presence (Def. 2.4.1), and for $\lambda = \frac{\Delta^A f}{\varepsilon}$, we obtain an ε-DP mechanism w.r.t. attribute A (Def. 2.4.3).

2.4.3. Qualitative Properties of Differential Privacy

Having presented the formal definition of differential privacy, below we discuss its qualities over other PETs [6]. Differential privacy stands out among other PETs because it protects against arbitrary attacks and moves beyond re-identification attacks protection. In this respect, the released event log would be almost the same, whether or not a given trace is included, and there is an obstacle for an attacker to detect the presence or absence of an individual, or to point out the activity instance(s) corresponding to an individual. Furthermore, differential privacy mitigates linkage attacks, where an attacker uses their background on the individual or the process to single out individuals. That applies to past, present, and future releases of event logs and additional sources of information that may help the attacker to perform a linkage attack.

Moreover, differential privacy provides quantification for privacy loss. This permits comparing different differentially private algorithms: given a fixed privacy bound, which algorithm ensures tighter privacy? Also, differential privacy enables quantification of privacy loss over repetitive access of the event log, which is called *composition*. That gives a flexibility to design complex approaches. For example, as we mentioned earlier in this thesis, an event log may contain many sensitive attributes and quasi-identifiers. Building a differentially private mechanism that protects all these attributes at the same time would be very complicated. The beauty of differential privacy is that it enables us to use a simpler differentially private mechanism for each attribute, and then, we can combine these mechanisms as building blocks of a bigger mechanism that still holds the differential privacy guarantees.

Besides, differential privacy is secure against any post-processing that can be performed by an attacker after releasing the anonymized data. In other words, the attacker cannot compute a function of the differentially private output that makes it less differentially private. Also, no matter any additional auxiliary information is available, the attacker cannot make the released event log less differentially private.

2.5. Secure Multi-party Computation (M3)

In some settings, organizations need ways of performing the computation in a private way and without sharing their private event logs. This can be achieved using secure computation protocols. Secure computation can be performed using outsourced computation and multi-party computation.

In the outsourced computation model, the event log owners outsource the computation process to a third party. The event log owner transforms their logs to an encrypted form that enables certain computations. The encrypted log is then shared to the third party to perform the computation. The third party can only perform computations without learning anything about the input log. At the end, the third party returns the output of the computations to the event log

owner, who can decrypt the results and get their private values. One protocol that enable such a computation is the *homomorphic encryption* [36, 37]. Homomorphic encryption (HE) can be used to perform computations (mostly linear) on event logs without seeing it [38].Any computations on encrypted event logs are enabled by *fully homomorphic encryption* (FHE) schemes [39]. While FHE-based approaches minimize the communication between parties, they are very computation-intensive.

Secure Multi-party Computation(MPC) [40] is a cryptographic functionality that allows n parties to cooperatively evaluate $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ for some function f, with the i-th party contributing the input x_i and learning the output y_i, and no party or an allowed coalition of parties learning nothing besides their own inputs and outputs. There exist a few approaches for constructing MPC protocols. Using the inherently 2-party *garbled circuits* (GC) approach [41, 42], one of the parties encrypts each gate of the Boolean circuit representing f, and sends it to the other party, together with the keys corresponding to both parties' inputs. The second party decrypts a part of the representation of each gate, and learns the output of f in the end. Garbled circuit based protocols have small round complexity, but tend to require more bandwidth than the approaches discussed below.

Homomorphic secret sharing [43] is a common basis for MPC protocols. In such protocols, the arithmetic or Boolean circuit representing f is evaluated gate-by-gate, constructing secret-shared outputs of gates from their secret-shared inputs. Each evaluation requires some communication between parties (except for addition gates), hence the depth of the circuit determines the round complexity of the protocol. On the other hand, there exist protocols with low communication complexity [44], allowing the secure computation of quite complex functions f, as long as the circuit implementing it has a low multiplicative depth.

In the secure multi-party computation (MPC) model, independent organizations can jointly perform computations on their input event logs without trusting a third party or sharing their logs. One difference between MPC and HE is that in MPC, all the input parties participate in executing the protocol.

The complexity of MPC protocols is dependent on the number of parties jointly performing the computations. Hence, the typical deployment of MPC has a small number of compute nodes, also known as *computation parties*, which execute the protocols for evaluating gates, while an unbounded number of parties may contribute the inputs and/or receive the outputs of the computation. Several frameworks support such deployments of MPC and provide APIs to simplify the development of privacy-preserving applications [45]. One of such frameworks is Sharemind [46], whose main protocol set is based on secret-sharing among three computing parties. In Chapter 7, we build on top of Sharemind, but our techniques are also applicable to other secret sharing-based MPC systems.

In Sharemind, a party can play different roles: an input party, a computation party, and/or an output party. In the case where only two parties are involved in an interorganizational process, these two parties play the role of input parties and also that of computing parties. To fulfill the requirements of Sharemind, they need to

enroll a third computing node, which merely performs computations using secret shares from which it can infer no information.[2]

The Sharemind framework provides its own programming language, namely the SecreC language [47], for programming privacy-preserving applications. SecreC allows us to abstract away certain details of the SMC cryptographic protocols used in Sharemind. In Chapter 7, we leverage SecreC to implement a technique for secure multi-party computation over event logs distributed across multiple organizations.

[2]When three or more parties are involved in a process, no external party is required.

3. PRIVACY-PRESERVING PROCESS MINING: REVIEW AND CONCEPTUAL FRAMEWORK

In this Chapter, we review the current state-of-the-art of prior research on privacy-preserving process mining. Also, we identify main threats, concepts, and methods in this field. This state-of-the-art review is not based on a systematic literature review, but rather on a compilation of related work in the field of privacy-preserving process mining gathered by a group of specialized researchers (the authors of reference [48]). We also develop a conceptual model that structures the discussion that existing techniques leave room for improvement. Specifically, this state-of-the-art review aims to answer the following questions:

Q1 What are the different threats for privacy and confidentiality in process mining?

Q2 What are the different requirements to achieve privacy-preserving process mining?

Q3 What approaches of privacy-enhancing technologies have been applied for process mining?

Q4 How could these approaches be classified?

Q1 aims to find the possible privacy threats and attacks on the published data in process mining. **Q2** aims to find different requirements based on data regulations (GDPR [2] in our case) that should be fulfilled to achieve privacy in process mining. **Q3** aims to identify the existing approaches of privacy-preserving process mining. **Q4** aims to define a taxonomy to classify the approaches based on the attacks they are able to mitigate, the requirements of the GDPR [2] that they fulfill, and the privacy models that they adopt. This Chapter is derived from [48] and contains sentences or fragments of sentences from this prior publication.

The remaining of the Chapter is structured as follows. Sect. 3.1 presents existing privacy threats in process mining and Sect. 3.2 presents different requirements to address these threats. Sect. 3.3 describes the existing approaches for privacy-preserving process mining. Sect. 3.4 summarizes the relevant studies.

3.1. Threats for Privacy and Confidentiality in Process Mining

This section presents threats to confidentiality and privacy in process mining. We collected the threats based on, first, developing a list of concrete attacks among the authors (this is a collaboration work presented in [48]) and, then, cross-referencing this list with generic attacks from the literature.

In this thesis, we do not focus on attacks with malicious adversaries that control the information flow. We assume an honest-but-curious attacker who follows the protocol to access the data for process mining and has legitimate access to the data. An attacker might be the process analysts or business user who obtains sensitive

information from supposedly anonymized event data provided to her. Inside or outside attacks to bypass access control are outside our scope.

Overall, the identified privacy threats can be categorized into four categories: re-identification, reconstruction, membership disclosure, cryptanalysis; each of which is described in more detail based on the literature and is instantiated in the context of the above hospital scenario.

3.1.1. Re-identification Threats (T1)

Re-identification or deanonymization threats are those where the identity of an information (data) subject is at risk to be disclosed by singling-out individuals from the supposedly anonymized event logs [49]. This threat of re-identification is currently most dominant at the data layer. Here an information subject may be directly linked to the process case, e.g., the patient in Table 1, or linked to a certain activity, e.g., a resource (employee) of the hospital. There are several possible attack methods described in the literature that constitute re-identification threats.

In a *linkage attack*, an attacker uses background or context knowledge on the process or on the individuals and combines it with the released artifact. A pseudonimized event log in the data layer of the process mining application may be such an artifact to which an analyst, or the general public in case of research data, has access. At first glance, Table 1 does not allow to re-identify patients or employees, all direct identifiers have been replaced. However, based on equal unique attributes, events can be linked to a case and thus to an identity. For instance, assume an attacker knows that Lena is 32 years old and has visited the hospital in a certain timeframe. By linking this very basic information with Table 1 only patient 3 is 32 years old, an attacker can infer the corresponding events of this patient and the sensitive disease. Such linkage attack can also be based on unique combinations of activities that are performed in a sequence for a certain process case. For example, the process case for the patient with identifier 2 is the only one containing two occurrences of the activity *Blood Test*. Thus, the uniqueness of cases in an event log can indicate the risk of re-identification. It was shown in [50] that there is serious potential for privacy leaks in published event log data, as the vast majority of public research event logs contain many unique cases.

When several organizations independently release generalized event logs about overlapping populations, re-identification is possible by an *intersection attack*. This may happen in the interorganizational process mining setting as illustrated with the benchmarking use case in our hospital example or, e.g., when several government agencies release event log data, which is likely to be containing information about the same population of citizens. If we assume that an adversary knows that a target is contained in several event logs, the identity may be disclosed by taking the intersection. Assume two hospitals independently publish event logs of patient trajectories in which age is generalized to prevent a linkage attack, i.e., Table 1 would only contain age groups 0–20, 21–40, and so on. Patient Lena would not be

easily re-identifiable anymore. However, let us assume that an adversary knows that she was transferred from hospital A to hospital B. Even though the age group is generalized, there may only be one process case in that age group in each of the hospital event logs such that it is consistent with the transfer scenario. So, the intersection set is a single record, and we have re-identified Lena. Of course, such an attack may also be performed on other event log attributes and not only be based on the time relation between process cases.

3.1.2. Reconstruction Threats (T2)

The threat of reconstruction is the risk of recreating the (partial) original event log from a released process model or aggregated statistics [51]. Reconstruction is a threat to privacy when attributes from individuals are reconstructed, and a threat to confidentiality when reconstructing non-personal data. This threat is closely linked to the presentation layer of the process mining application. The individuals in the event log are seemingly protected by only releasing aggregate statistics.

If an individual cannot be linked directly, attributes can be reconstructed, for example, from aggregated statistics by a *difference attack*. In this attack, an adversary isolates a single value by combining multiple aggregated statistics about an event log. Assume the process analyst can only pose queries to obtain aggregate statistics, e.g., by obtaining process model visualizations such as the one in Figure 2. The analyst could obtain the frequency visualization grouped per disease and, therefore, know the number of patients (unique cases) for each disease. In a second query, the analyst could exclude 32-year-old patients in the query and obtain the same statistics. From the difference, an adversary can infer that Lena, the only 32-year-old patient, has an infection.

Adversaries may also attempt to reconstruct training or source data from a published model, which is called *model-inversion* [52]. The training data is estimated by observing the input and output of a model. This attack only creates a probabilistic version of the training data. The models used for predictive and prescriptive process monitoring [53, 54] use machine learning models that could be directly vulnerable to this attack. In our application scenario, however, even a probabilistic version of the original event log can reveal private information such as the diagnosis of a specific patient or the responsible resources treating a patient. From the process model in Figure 2, we can infer that the underlying training event log consists of a single trace in which the patient left without being hospitalized and also that there is a single patient that had two blood tests taken.

3.1.3. Membership Disclosure Threats (T3)

Membership disclosure threats entail uncovering the knowledge of whether a specific individual was included in the source data for a particular model or analysis. So, differently to a re-identification attack, only the fact that an individual was part of the event log is disclosed.

In a *membership inference attack*, an adversary aims to determine whether an individual was included in the source data. An adversary who only has access to the released model can train shadow models to predict membership [55]. These models capture the misclassification difference between samples that are likely to be in or outside the training data. By checking if a process model allows the behavior of a certain trace, an adversary can try to predict, if the trace was included in the data underlying the process discovery. In our hospital example, depending on the scope of a process mining analysis, such knowledge could reveal that a specific individual visited the hospital for a specific treatment. For instance, assume an attacker knows that a target patient Anna has made a blood test twice and has access to the process model (cf. Figure 2). It is very likely that Anna is included, since he made a blood test twice. However, in this case, an adversary cannot identify that Anna is patient 2, i.e., the exact process case identifier remains protected. Still, this knowledge may leak sensitive information, e.g., if the event log was extracted for a set of patients with a specific disease membership disclosure would also disclose the disease information.

3.1.4. Cryptanalysis Threats (T4)

Often, data is pseudonymized, as in our example, or even encrypted to provide confidentiality. However, pseudonymized or even fully encrypted event logs are vulnerable to attacks based on the analysis of the frequency.

A *frequency analysis* takes advantage of the characteristics of the encrypted data. Such analysis could rely on background knowledge of the process, e.g. the frequency of certain activities within one trace and their position in the trace. For example, when considering our hospital process example, certain diagnostic steps, such as blood test, might appear more than once in a trace, while the registration and release of a patient usually happen at the beginning and end of each trace. In other words, even if the activity in Table 1 had been encrypted, an attacker might decode the start and end activity. In this case, an attacker has both the plaintext and its encrypted version, which can be used to reveal the total cipher. It is not difficult to gain this knowledge, especially in public places like a hospital.

We identified four main threats and sketched the related attack methods in a process mining scenario. In the remainder of this Chapter, we review and discuss possible solutions and identify the open challenges.

3.2. Conceptual Model and Requirements of PPPM

In this section, we discuss requirements that address the threats to confidentiality and privacy in process mining, discussed in Section 3.1. The requirements are taken from a systematic synthesis of the current privacy and confidentiality landscape conducted by Gharib et al. [56], who themselves based their work on a previous literature review [57]. The mentioned requirements are legislature agnostic but

nonetheless present the opportunity to incorporate demands and elements of multiple common protection models such as the European (GDPR), Australian (Privacy Act 1988), Canadian (PIPEDA), and US legislation. We will particularly focus on GDPR as an example to explain the origin of the requirements.

Figure 3 shows a UML class diagram of how the concepts of privacy and confidentiality relate, both to the discussed threats and to the requirements taken from [56]. In Section 2.1, we note that confidentiality has direct overlap with privacy. Therefore, confidentiality could, as in Gharib et al. [56], solely be seen as a requirement of privacy. However, following our definition in Section 2.1, we raise confidentiality beyond just its overlap with privacy and make it a separate concept, therefore slightly adjusting the model in Gharib et al. [56]. The purpose of our privacy and confidentiality requirements is to address potential threats and mitigate potential vulnerabilities. In the following, we explain the requirements.

- **R1 - Anonymity** deals with personal information, and it ensures that personal data can only be used without disclosing identities of information subjects [58, 59]. This is a privacy requirement since it concerns personal information. According to Recital 26 of GDPR, the principles of data protection should not apply to anonymous information where the information subject is not identifiable. Therefore, anonymity is a big step towards privacy regulations compliance.
- **R2 - Unlinkability** describes that it should not be possible to link personal information to their corresponding owners [59]. This requirement complements R1 in the sense that preventing identity disclosure cannot be guaranteed by only making identifiers unreadable. Furthermore, all identifiers for linkage also need to be removed. Unlinkability includes that data cannot be re-identified by linkage attacks.
- **R3 - Unobservability** means that it should not be possible to observe the identities of information subjects that perform any action [59]. It should be noted that unlike anonymity and unlinkability, which keep the identity of the actor hidden, the goal here is to ensure that the actions themselves are hidden. Thus, we categorize this requirement as a confidentiality requirement. This requirement mostly concerns continuous monitoring and processing of the data generated by running systems. It is intended to protect personal data against unauthorized processing, as described in Article 5 of GDPR.
- **R4 - Notice** means that information subjects should be notified when their information is gathered [58]. A notice should include the detailed information which is to be gathered, disclosure risks and data quality concerns. The notice requirement can be categorized as both privacy and confidentiality requirement depending on the information owner, which could be both an individual or a company. This requirement relates to the concept of *consent* in GDPR. The data processing often needs to be based on consent. As mentioned in Article 7 of GDPR, the data controller/processor should

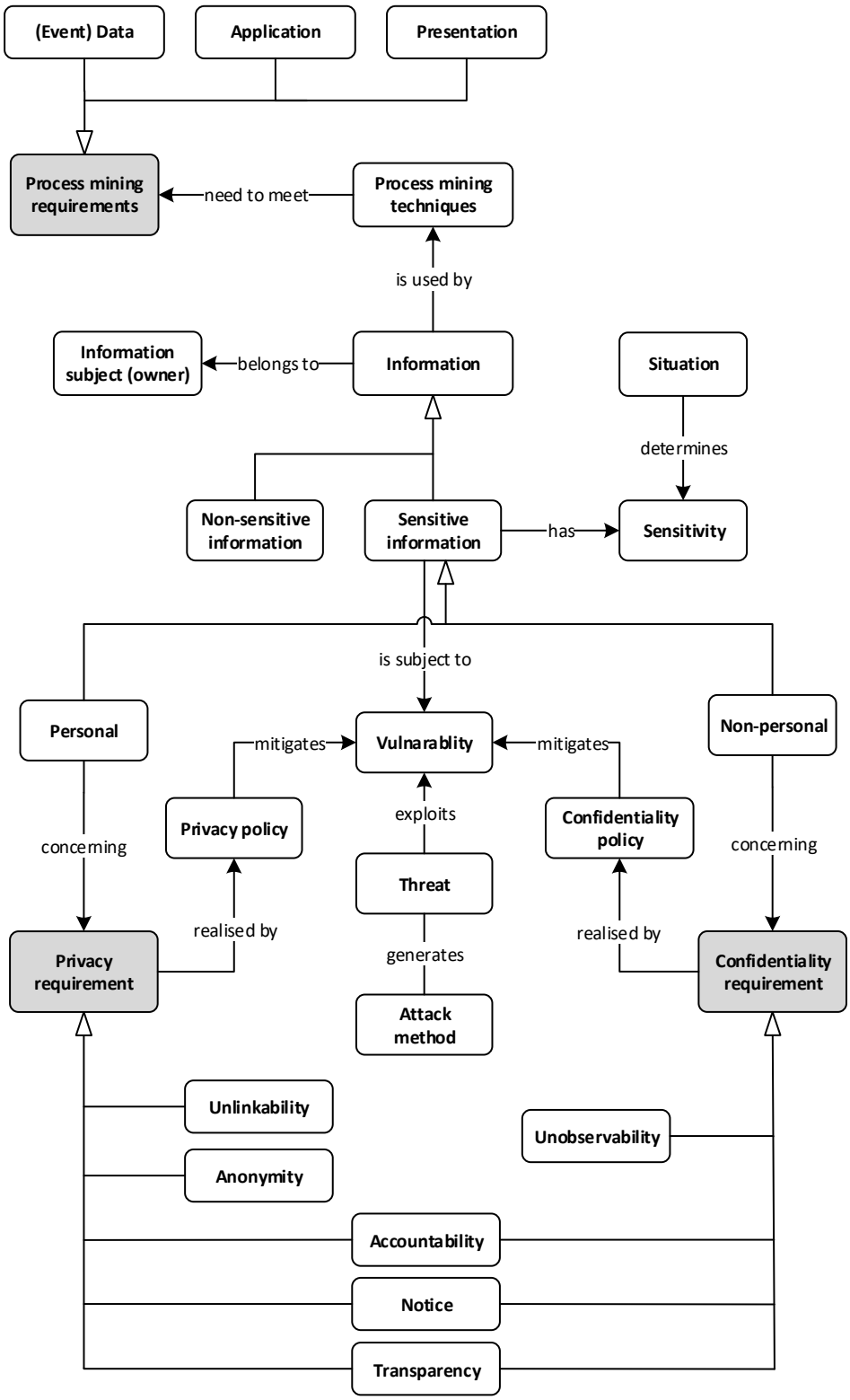


Figure 3. A UML class diagram linking threats and requirements to the remaining concepts used.

demonstrate that the information subject has consented to processing of his/her data. Note that the consent needs to be kept updated based on the purpose of data processing.

- **R5 - Transparency** means information owners should be able to know who uses their information, how, and for what purposes [58]. This requirement can also be categorized as both privacy and confidentiality requirement, depending on the nature of the information owner. The principle of transparency is described in Recital 58 of GDPR.
- **R6 - Accountability** describes that information subjects should be able to hold information users accountable for their activities and the consequences of misusing their data [58]. This requirement concerns both personal and non-personal information. It is mentioned in Article 5 GDPR as one of the principles of processing personal data, where it is described that the personal data should be protected against unlawful processing, accidental loss, destruction, or damage.

When information is exploited by process mining techniques, protecting privacy and confidentiality should not compromise the requirements of process mining techniques. There are three types of additional requirements:

- **R7 - Data** requirements are that process mining techniques should support different data storage formats, e.g., centralized in a single organization and distributed among different parties. On top of that, processing different types of data should not lead to any privacy leakage.
- **R8 - Application** requirements mean that the algorithms which are applied should be computationally acceptable, and fulfilling privacy and confidentiality requirements should not impose an unreasonable load on the time or resource consumption of the algorithms.
- **R9 - Presentation** requirements are that the reported results should be interpretable by users. This includes fulfilling privacy and confidentiality requirements without leading to a utility loss of the anonymized data, and having the ability to repeat different types of queries without privacy disclosure.

3.3. Existing Approaches for Privacy-Preserving Process Mining

Previous studies on privacy-preserving and confidential process mining addressed RQ1 in the light of the requirement UR2 and attempts to mitigate the above threats. In order to achieve the privacy in process mining, an approach needs to achieve privacy from the following perspectives [60]:

- Control flow perspective, which represents the traces of individuals under study. E.g., in a health care system, we need to anonymize the traces of patients and their activities.

- Organizational perspective, which focuses on the individuals who execute the activities within the organization, i.e., resources. For example, in a health case environment, doctors and nurses execute the activities.
- Case perspective, which focuses on the case attributes that can link a case to its individual.
- Time perspective, which is represented as the timestamp at which an activity has been executed.

Below, we list different approaches that implement PETs for process mining.

3.3.1. Group-based Approaches

Group-based models (M1) have been addressed in several studies [12, 60–63]. Rafiei et al. [12] address RQ1 and propose a privacy model called TLKC for publishing event logs, which provides group-based anonymization (M1) based on k-anonymity, and quantifies the risk based on the attacker's background knowledge. This work anonymizes the event log from the control flow, time, and case perspectives. Their model fulfills privacy requirements R1 - Anonymity and R2 - Unlinkability. The k-anonymity model is secure against the reconstruction threat (T2). However, k-anonymity partially mitigates the re-identification threats (T1) and the membership disclosure threat (T3) [4]. The adoption of k-anonymity makes the model interpretable (R9 - Presentation). Although, TLKC results in suppressing cases or events within cases. In one example in [12], TLKC suppressed 87% of the activities in the output. Accordingly, this study does not fulfill utility requirement **UR1a** (as explained earlier in Sect. 1.1) because the case variants in the anonymized log are different from the original log due to the suppression of cases or events within cases. Also, it does not fulfill **UR1b** because the suppression of events within cases results in traces that do not exist in real-life.

Rafiei et al. [60] extended the above work to provide log anonymization approach from the organizational perspective. The approach in [60] identifies traces that have higher risk than the others. Later on, the approach uses the quantified risk and a threshold k to suppress the violating traces and map them to non-violating traces. This approach addresses RQ1 under the utility requirement **UR1b** because it does not introduce new case variants to the anonymized event log.

Also, the authors in [61, 62] use the k-anonymity privacy model (M1) for publishing event logs and add the use of t-closeness to protect attribute values. They provide privacy from the organizational perspective, and hence, they do not address RQ1. Similar to TLKC, R1 - Anonymity, R2 - Unlinkability requirements are fulfilled. Again, the privacy model is interpretable (R9 - Presentation). The authors suppress events or cases that would result in privacy leakage. However, data suppression is correlated with utility loss and no explicit utility measure is considered. PRETSA prunes entire traces and identifies the traces that are most similar to the pruned once to replace them. Because of that, PRETSA does not fulfill **UR1a**, but it fulfills **UR1b**.

Batista and Solanas [63] present an approach that anonymize the event log from the organizational perspective. The approach adopts a group-based privacy model that groups individuals based on activity distribution and transfers resource information within the group to uniform the resource distribution. The transfer of activities within the groups of individuals does not fulfill the utility requirements **UR1a** and **UR1b**. Also, this approach does not address RQ1. Similar to the above approaches, this approach fulfills R1 - Anonymity, R2 - Unlinkability requirements, and R9 - Presentation. In addition, this approach is secure against the reconstruction threat (T2), but it only partially mitigates the re-identification threats (T1) and the membership disclosure threat (T3) [4].

3.3.2. Differential Privacy Approaches

The differential-privacy model (M2) is adopted, e.g., by Mannhardt et al. [64] for controlling disclosure of two types of queries: the frequencies of directly-follows relations and the case variant frequencies. Therefore, this approach does not address RQ1. Mannhardt et al. [64] anonymizes the data from the control flow perspective. Re-identification threats T1, reconstruction threats T2, and membership disclosure threats T3 are mitigated and requirements R1 and R2 are fulfilled. Neither transparency nor accountability is considered, and no guarantees are given for the utility (R9 - Presentation) of the obtained DFG. This work does not fulfill the utility requirements **UR1a** and **UR1b** because it suppresses activities within traces which introduces case variants that do not exist in real life and suppresses case variants that exist in the original data. Also, this approach does not consider logs with timestamps.

PRIPEL [65] uses timestamp shifts to anonymize the timestamp attribute of the log. It ensures privacy guarantees based on individual cases. Also, PRIPEL uses sequence enrichment to anonymize other attributes of the log. PRIPEL takes three input parameters, namely ε, k, and N. ε is the DP parameter, k is the cut-out frequency (i.e., PRIPEL cuts out all variants that appear less than k), and N is the maximal prefix length. The use of differential privacy fulfills the requirements R1 - Anonymity and R2 - Unlinkability to mitigate threats T1, T2, and T3. However, PRIPEL does not optimize the disclosure for a certain level of utility (R9 - Presentation). The output of the above approaches [64,65] may contain variants that were never observed in the event log. Besides, some newly injected variants are impossible to occur for the anonymized process [66]. Hence, this work does not fulfill the utility requirements **UR1a** and **UR1b**. Furthermore, the injected noise over the timestamp column is not minimal, so it partially addresses RQ1.

SaCoFa [66] is another differentially private mechanism that achieves lower utility loss than the above approaches by means of semantic constraints. This approach adopts differential privacy to replace prefixes that are common in multiple cases with perturbed ones, given that the distance between the perturbed prefixes and the original one does not exceed a certain distance. SaCoFa achieves the

privacy from only the control flow perspective and partially addresses RQ1. This approach fulfills the requirements R1 - Anonymity and R2 - Unlinkability to mitigate threats T1, T2, and T3. Like the above two approaches, this approach suffers from the existence of false negatives (traces that do not happen in the event log) and false positives (traces that happened in the original log but do not appear in the anonymized log). And hence, this work does not fulfill the utility requirements **UR1a** and **UR1b**.

3.3.3. Cryptographic privacy models

Cryptographic privacy models (M3) have been considered for both centralized and distributed event logs settings. Rafiei et al. [67, 68] introduced an encryption framework for ensuring confidentiality in process mining to secure against the threat T4. The framework is divided into three processing environments and provides user access control, thereby addressing R6 - Accountability. The framework gives a data analyst access to the internal partially secure event logs, which makes the framework vulnerable against T1, T2, and T3. Only centralized event logs are considered failing to provide support for distributed event logs (R7 - Data). This approach does not address RQ2.

Tillem et al. [69] propose a secure processing protocol to execute the Alpha algorithm over distributed event logs (R7 - Data) in a cross-organizational setting. This approach targets solely the creation of a process model. Also, Tillem et al. [69] requires a trusted third-party between the collaborating organizations, which makes their approach does not fully address RQ2. However, their protocol does not mitigate attacks on confidentiality described in T4.

3.3.4. Other studies on Privacy-Preserving Process Mining

Other studies on privacy-preserving process mining neither address RQ1 and RQ2 nor fulfill any of the above requirements, as they do not provide a concrete mechanism of disclosure control. Rafiei et al. [70, 71] provide privacy metadata by extending the XES standard. Pika et al. [72] studied the impact of anonymization on process mining in healthcare without providing a concrete mechanism. In this line, Rafiei et al. [73] provide privacy quantification for both the disclosure risk and the utility loss and Nuñez von Voigt et al. [50] quantify the re-identification risk resulted from the disclosure of event logs based on individual uniqueness. Both do not provide a solution. Maatouk et al. [74] provides a quantification of the re-identification risk in published process models.

Kabierski et al. [75] provide a framework for anonymizing process performance indicators. Rösel et al. [76] present a distance measure based on feature learning to merge traces during event log anonymization. Other studies addressed the privacy-preserving continuous event data publishing [77]. Zaman et al. [78] studied the GDPR compliance in business processes through data-driven solutions. An intra-process data degradation in business processes has been proposed in [79]. Finally,

other works offer only one specific task, for example, Rafiei et al. [80] provide privacy-preserving role mining adopting a substitution method that secures the activities with sensitive frequencies.

3.4. Summary

We summarize the most relevant studies in Table 3 and observe that some of the group-based approaches address RQ1 under the utility requirement UR2, but they do not mitigate PSO attacks. Further, the above differentially-private mechanisms do not address RQ1 under UR1 nor UR2. Also, none of the above approaches address RQ2.The studies in Sect. 3.3.4 are not included as they do not address any of the research questions.

Most of the previous studies fulfill the requirements: R1, R2, and R3. However, some of the requirements have not been addressed in the literature, e.g., R4, R5, R8, and R9. Furthermore, the literature either mitigates threats on privacy or threats on confidentiality, but it does not mitigate both types of the attacks together.

Table 3. Summary of privacy-preserving process mining approaches w.r.t. the requirements, the protection models, and the threats (the symbol ✓means fulfillment, $*$ means partial fulfillment, - means does not fulfill, and NA stands for not applicable)

Paper	R1	R2	R3	R4	R5	R6	R7	R8	R9	M1	M2	M3	T1	T2	T3	T4	RQ1	RQ2	UR1a	UR1b	UR2
TLKC [12]	✓	✓	-	-	-	-	-	-	✓	✓	-	-	*	✓	*	-	*	-	-	-	-
TLKC extension [60]	✓	✓	-	-	-	-	-	-	✓	✓	-	-	*	✓	*	-	*	-	-	✓	-
PRETSA [61]	✓	✓	-	-	-	-	-	-	✓	✓	-	-	*	✓	*	-	-	-	-	✓	-
Batista and Solanas [63]	✓	✓	-	-	-	-	-	-	✓	✓	-	-	*	✓	*	-	-	-	-	-	-
Mannhardt et al. [64]	✓	✓	-	-	-	-	-	-	-	-	✓	-	✓	✓	✓	-	-	-	-	-	-
PRIPEL [65]	✓	✓	-	-	-	-	-	-	-	-	✓	-	✓	✓	✓	-	*	-	-	-	-
SaCoFa [66]	✓	✓	-	-	-	-	-	-	-	-	✓	-	✓	✓	✓	-	*	-	-	-	-
Rafiei et al. [67, 68]	-	-	✓	-	-	✓	-	-	-	-	-	✓	-	-	-	✓	-	-	NA	NA	NA
Tillem et al. [69]	-	-	-	-	-	-	✓	-	-	-	-	✓	-	-	-	-	-	*	NA	NA	NA

4. DIFFERENTIALLY PRIVATE EVENT LOGS: AN OVERSAMPLING APPROACH

In this chapter, we address RQ1 with the guessing advantage as our privacy risk threshold. In other words, in this chapter, we address the question *Given an event log L, and given a maximum level of acceptable guessing advantage δ, how to generate an anonymized event log L' such that the success probability of singling out an individual after publishing L' does not increase by more than δ.*

We do that under the utility requirement *UR1a: The anonymized event log must have the same set of case variants as the original log* and the utility requirement *UR2: The difference between the real and the anonymized time values is minimal given a privacy risk metric.*

In response, this chapter presents a differentially private mechanism that anonymizes event logs while keeping the case variants in the anonymized log the same as the original one. The mechanism determines the needed ε value to disclose an event log in terms of a business-related metric, namely guessing advantage, which captures the increase in the probability that an adversary may guess information about an individual after the disclosure. The probability of leakage is a widely used measure of risk which can be interpreted on its own.

Usually, an adversary has prior knowledge about the individuals in the log before its release. Therefore, the adversary gains additional information that allows them to guess personal information following the disclosure successfully. The goal of anonymization is to limit this risk. To this end, we define the maximum acceptable risk threshold as the maximum guessing advantage level δ, which is the difference in the probability that an adversary singles out an individual before and after releasing the anonymized event log.

Given a maximum allowed guessing advantage, δ, a differentially private event log is obtained by oversampling the traces in the log and injecting noise to the event timestamps. This ensures that the probability that an attacker may single out any individual, based on the prefixes/suffixes of the individual's trace or based on the event timestamps, is not more than δ.

The proposal relies on a data structure that compactly captures all prefixes and suffixes of a set of traces, namely a Deterministic Acyclic Finite State Automata (DAFSA) [14]. A DAFSA is a lossless representation of an event log, wherein every prefix or suffix shared by multiple traces is represented once. By analyzing the frequency and time differences of each DAFSA transition, we determine the amount of oversampling and timestamp noise. The estimated noise limits the guessing advantage of an attacker by inspecting the traces that traverse this transition in the anonymized log.

This chapter is structured as follows. Sect 4.2 presents the notion of the differentially private event log, the attack model, and the risk quantification approach. Sect. 4.3 translates the risk quantification into an algorithm that anonymizes a

log. Sect. 4.4 concludes the contribution. The evaluation of this contribution is presented in Sect.5.3.

This chapter is derived from [15] and contains sentences or fragments of sentences from this prior publication.

4.1. Attack Model

We consider a scenario where an organization shares its event log with an analyst, who should not be able to infer an individual's information. We assume that the activity labels and the smallest and largest timestamp in the event log are public information. We assume that each trace in the log pertains to an individual whose privacy we wish to safeguard. We view the analyst as a potential attacker who may seek to infer information about an individual based on the released log. We seek to protect the release under a worst-case scenario where the analyst has background knowledge about all individuals in the log except for the individual of their interest. Specifically, we seek to protect the release of log L to prevent the attacker from fulfilling one or both of the following attack goals

- h_1: Distinguishing whether an individual has participated in the log or not through their execution control flow.
- h_2: Determining the execution timestamp of an activity.

Note that we do not seek to prevent the attacker from guessing the activity labels, i.e., we do not view the activity labels as private information. Also, we assume that cases are independent, meaning that the sequence of activities that a case follows does not depend on the activity sequences of other cases. This assumption usually holds in a business process, e.g., the patient's pathway in a treatment process does not depend on that of other patients.

The above assumption might not hold for some business processes where an individual participates in more than one case in the event log. To estimate the differential privacy guarantees in such a setting, the privacy parameter ε shall be multiplied by the largest number of cases an individual can participate in.

To prevent an attacker from achieving goals h_1 and h_2, we introduce a notion of a differentially private event log.

Definition 4.1.1 (Differentially Private Event Log)**.** Let L be an event log as defined in Def. 2.2.1. We say that a log $M(L)$ is ε-differentially private if: (1) it ensures ε-differential privacy from the control-flow perspective; and (2) it is ε-differentially private w.r.t. timestamp.

4.2. Differential Privacy Mechanism

In this section, we introduce the concepts of DAFSA. We explain how to provide differential privacy guarantee to an event log notion, and risk quantification.

4.2.1. Event Log Representation

Table 4. Example of an event log

Case ID	Activity	Timestamp	Other Attributes
1	A	8/8/2020 10:20	. . . .
	B	8/8/2020 10:50	. . . .
	C	8/8/2020 16:15	. . . .
2	D	8/8/2020 12:37	. . . .
	A	8/8/2020 14:37	. . . .
	E	8/8/2020 15:07	. . . .
	C	8/8/2020 20:31	. . . .
3	A	8/9/2020 13:30	. . . .
	B	8/9/2020 13:55	. . . .
	C	8/9/2020 20:55	. . . .
4	D	8/9/2020 15:00	. . . .
	A	8/9/2020 17:00	. . . .
	B	8/9/2020 17:40	. . . .
	C	8/9/2020 23:05	. . . .
5	A	8/9/2020 17:25	. . . .
	E	8/9/2020 17:55	. . . .
	C	8/10/2020 23:55	. . . .
6	A	8/11/2020 17:00	. . . .
	B	8/11/2020 17:27	. . . .
	C	8/11/2020 23:45	. . . .

We assume that each trace in a log corresponds to the execution of the process pertaining to an individual whose privacy we wish to safeguard. Specifically, our goal is to mitigate singling out of an individual based on any prefix or suffix of their trace. To achieve this, we group the prefixes and suffixes in the log, and we inject independent differentially private noise to each such group. For this, we need a representation of the log that partitions the prefixes and suffixes of the log traces into groups. In other words, this representation should assign each prefix (suffix) in the log to one group such that the union of the groups is equal to the entire set of prefixes (suffixes). In addition, we also require that this representation preserves the set of case variants of the log (cf. **UR1a**). As a running example, we use the event log presented in Table 4.

The DAFSA provides such a partitioning. Given a set of words, each state in the DAFSA represents a group of prefixes that share the same set of suffixes, and suffixes that share the same set of prefixes [14].

An advantage of the DAFSA (specifically the *minimal DAFSA*) over similar representations, such as prefix trees, is that a (minimal) DAFSA contains a minimal number of groups (states) [14]. By minimizing the number of groups, we obtain larger groups. The larger the group, the smaller the needed noise injection to

achieve ε-DP.

Definition 4.2.1 (Minimal DAFSA of a set of words [14])**.** Let V be a finite set of labels. A DAFSA is an acyclic-directed graph $D = (S, s_0, A, S_f)$, where S is a finite set of states, $s_0 \in S$ is the initial state, $A \subset S \times V \times S$ is a set of labeled transitions, and S_f is a set of final states. A DAFSA of a set of words W is a DAFSA such that every word in W is a path from an initial to a final state, and, conversely, every path from an initial state to a final state is a word in W. A minimal DAFSA of a set of words W is a DAFSA of W with a minimal number of states.

Given a DAFSA constructed from a set of words, every word is a path from an initial state to a final state. Conversely, every path from an initial state to a final state corresponds to a word in the given set of words [14]. Reissner et al. [81] reuse the algorithm in [14] to represent a log as a DAFSA. Every trace is seen as a word (its corresponding case variant). For example, the DAFSA of the log in Table 5 is shown in Fig. 4.

Definition 4.2.2 (DAFSA [14])**.** Let V be a finite non-empty set of (activity) labels. A DAFSA is an acyclic and deterministic directed graph $D = (S, s_0, A, S_f)$, where S is a finite set of states, $s_0 \in S$ is the initial state, with a set of arcs $A \subset S \times V \times S$, and a set of final states S_f.

Given a path from the initial state S_0 to a state $s \in S$, the set of labels associated with the arcs in the path is referred to as the *prefix* of s. Similarly, given a path from s to the final state s_f, the set of labels associated with such path are referred to as the *suffix* of s. In other words, the set of prefixes of a state s can be represented as:

$$pref(s) = \bigcup_{(s_s, a, s_t) \in \blacktriangleright s} \{x \oplus a | x \in pref(s_s)\}, \tag{4.1}$$

where $\oplus$ represents concatenation, and $\blacktriangleright s$ is the set of incoming arcs of s. $pref(S_0) = \{\langle\rangle\}$. Similarly, the set of suffixes of a state s can be represented as

$$suff(s) = \bigcup_{(s_s, a, s_t) \in s \blacktriangleright} \{x \oplus a | x \in suff(s_t)\}, \tag{4.2}$$

where $s \blacktriangleright$ is the set of outgoing arcs of s. $pref(S_f) = \{\langle\rangle\}$. Prefixes and suffixes are called common iff they are shared by more than one trace.

Definition 4.2.3 (Common prefixes and suffixes [81])**.** Let $D = (S, s_0, A, S_f)$ be a DAFSA. The set of common prefixes of D is $\mathbb{P} = \{pref(s) | s \in S \wedge |s \blacktriangleright| > 1\}$. The set of common suffixes of D is $\mathbb{S} = \{suff(s) | s \in S \wedge |\blacktriangleright s| > 1\}$.

The common prefixes of the DAFSA in Fig. 4 are $\{\langle A, B\rangle \langle D, A\rangle \langle A\rangle\}$, and the common suffixes are $\{\langle B, C\rangle \langle E, C\rangle\}$. Cases corresponding to case variants that traverse a given DAFSA state s share the same set of prefixes and suffixes. In this chapter, we employ DAFSA states and transitions to group common prefixes and suffixes within cases of the log. We annotate the log with the DAFSA transitions and states to relate such grouping to the event log cases.

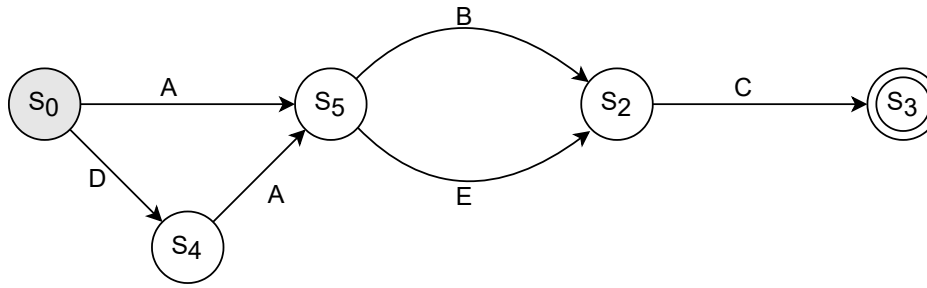


Figure 4. DAFSA of the event log in Table 5

Definition 4.2.4 (State Annotated Event Log)**.** A state annotated event log $L_s = \{r_1, r_2, ..., r_n\}$ is a set of entries $r = (i, a, ts, s_i, s_e)$, each links an event $e = (i, a, ts) \in L$, where L is the event log, to the DAFSA transition $t = (s_i, a, s_e)$ that represents the occurrence of that event, where t starts from s_i, ends at s_e, and labeled with the same activity a.

The state-annotated event log links every event in the event log, based on its prefix, suffix, and activity label, to the DAFSA transition. Every event is labeled by the source state and target state of the DAFSA transition. Table 5 (columns source state and target state) shows the state annotated event log.

4.2.2. Privacy Mechanism

Given Definitions 4.2.2, and 4.2.4, all cases that share a common prefix will traverse a given state s in the DAFSA corresponding to this prefix. The same holds for cases that share a common suffix. We quantify the privacy parameter ε using the state annotated event log to mitigate singling out an individual by their prefixes (or suffixes). To study the histogram (count) distribution of the common prefixes and suffixes between traces of an event log, we construct the *DAFSA transition contingency table*. A contingency table is a histogram of group occurrences.

Definition 4.2.5 (DAFSA Transitions Contingency Table)**.** The DAFSA transition contingency table C is the histogram of counts for each transition $t = (s_i, a, s_e)$ of the DAFSA D, where s_i is the source state, a is the activity label, and s_e is the target state of t.

Table 6 shows the DAFSA transitions contingency table. These counts are usually called marginals. The marginals contain the correlations counts of the common sets of prefixes and suffixes of the DAFSA. We anonymize these marginals to prevent singling out that an individual has been through an activity, using the prefix and the suffix set of activities. The anonymization of the marginals can be directly mapped to the privacy-preserving Online Analytical Processing [82].

Definition 4.2.6 (Differentially Private DAFSA Transitions Contingency Table)**.** Let f be a query function that computes a DAFSA transitions contingency table that has a set of transitions $t = (s_i, a, s_e)$ and a count cell c_i for each transition. Let M_f be an unbounded ε-differentially private mechanism (by Def. 2.4.1) that

Table 5. DAFSA State-Annotated Event log

Case	Activity	Timestamp	Source State	Target State
1	A	8/8/2020 10:20	s_0	s_5
	B	8/8/2020 10:50	s_5	s_2
	C	8/8/2020 16:15	s_2	s_3
2	D	8/8/2020 12:37	s_0	s_4
	A	8/8/2020 14:37	s_4	s_5
	E	8/8/2020 15:07	s_5	s_2
	C	8/8/2020 20:31	s_2	s_3
3	A	8/9/2020 13:30	s_0	s_5
	B	8/9/2020 13:55	s_5	s_2
	C	8/9/2020 20:55	s_2	s_3
4	D	8/9/2020 15:00	s_0	s_4
	A	8/9/2020 17:00	s_4	s_5
	B	8/9/2020 17:40	s_5	s_2
	C	8/9/2020 23:05	s_2	s_3
5	A	8/9/2020 17:25	s_0	s_5
	E	8/9/2020 17:55	s_5	s_2
	C	8/10/2020 23:55	s_2	s_3
6	A	8/11/2020 17:00	s_0	s_5
	B	8/11/2020 17:27	s_5	s_2
	C	8/11/2020 23:45	s_2	s_3

Table 6. DAFSA Transitions Contingency Table

Source State	Activity	Target State	Count
s_0	A	s_5	3
s_5	B	s_2	3
s_2	C	s_3	5
s_0	D	s_4	2
s_4	A	s_5	2
s_5	E	s_2	2

injects noise into the result of f. The differentially private DAFSA transitions contingency table is: $M(C) := \{(t_1, M_f(c_1)), (t_2, M_f(c_2))..., (t_n, M_f(c_n))\}$, where n is the number of DAFSA transitions.

Our goal is not to anonymize a DAFSA, but an *event log*. We assume that case IDs have been pseudonymized, the activity labels are public, and the individual cases are independent. We propose a mechanism M that anonymizes two properties of an event log: the activity timestamp and the set of prefixes and suffixes of activity. We can release the time for the activity timestamp attribute by applying a bounded ε-DP mechanism w.r.t. timestamp attribute (Def. 2.4.3). We need a mechanism that makes the contingency table ε-DP (Def. 4.2.6) for the second property.

The mechanism M_f of Def. 4.2.6 operates on counts and cannot be applied to a log directly. We need to translate the noise injection of the contingency table to the log. Kifer et al. [83] present the notion of a *move* as a way of anonymizing the marginals of contingency tables. A move is a process that adds or deletes a tuple from the contingency table. In order to fulfill the requirement **UR1a**, we define *oversampling* as increasing a count (positive move) in the contingency table by replicating a random tuple in the log.

Definition 4.2.7 (DAFSA Transition Oversample)**.** Given a DAFSA transition contingency table C_i, an oversample O is a transformation that adds a DAFSA transition instance to C_i, producing a contingency table $C_j = O(C_i)$, with an increase of only one count cell by 1.

We define a mechanism M that oversamples tuples of a log L so that, if we computed the contingency table of $M(L)$, it would be ε-DP. We must be careful that $M(L)$ does not leak anything that the contingency table would not leak. For example, if we keep timestamps unchanged, it suffices to remove the tuples with repeating timestamps to eliminate duplicates, so we need to make the times ε-DP as well. Since duplicated timestamps are correlated, we have to divide the ε values of the time queries of the replicated cases by their oversampling ratio. To keep the counts in the DAFSA transitions consistent, we oversample the prefix and the suffix of the oversampled transition, i.e. we always oversample entire cases.

Definition 4.2.8 (Case Oversample)**.** Given an event log L, a case oversample O_c is a transformation that duplicates a case c_i of log L, in such a way that c_j, the duplicated case in log $O_c(L)$, and c_i have the same sequence of activities.

4.2.3. Risk Quantification

We use an ε-differential privacy mechanism to mitigate the attacker's goals h_1 and h_2. To determine the privacy parameter ε, we adopt the guessing advantage framework [84], which quantifies the risk of publishing a dataset as the difference between two probabilities: the prior and the posterior guessing probability. Even without publishing the event log, attackers can use their knowledge to guess information about a specific individual. The guess is considered successful if it

falls within a range of values H_p, which is the actual value, $\pm$ a precision.

Definition 4.2.9 (Prior Guessing Probability)**.** An attacker's prior guessing probability is defined as $P := Pr[h(L) \in H_p]$

A guessing precision p is a percentage value representing the range of a successful guess H_p. For example, if the time between two executive events of the same trace in a log is 0.5 hour, and $p = 0.2$ (12 minutes), the guessed value is considered successful if it falls in range $H_p = [0.3..0.7]$ hour, i.e., [18 .. 42] minutes. Our method pre-processes the log to normalize the values (relative timestamps) to be between 0 and 1. The precision is interpreted within this [0,1] range.

The guessing advantage is defined as the difference between the posterior probability (after publishing $M(L)$) and the prior probability (before publishing $M(L)$) of an attacker making a successful guess in H_p. Let δ be the maximum allowed guessing advantage, stated by the event log publisher.

Definition 4.2.10 (Guessing Advantage)**.** Attacker's advantage in achieving the goal h with precision p is at most δ if, for any published event log L',

$$Pr[h(L') \in H_p \mid M(L) = L'] - Pr[h(L) \in H_p] \leq \delta.$$

Laud et al. [84, 85] proposed estimation of the posterior guessing probability.

Proposition 4.2.1 (Posterior Guessing Probability [84,85])**.** *The posterior guessing probability of an attribute ranging between 0 and r for a single individual after the release of the timestamp attribute of an event log is bounded by*

$$P' \leq \frac{1}{1 + exp(-\varepsilon \cdot r)\frac{1-P}{P}} \quad .$$

Proof. (Taken from [84, 85]) An attacker has a prior knowledge $k(l)$ of part of the event log l. Using the equality $Pr[X = x] = \sum_{y \in Y} Pr[X = x, Y = y]$ and Bayesian formula $Pr[A,B] = Pr[A|B] \cdot Pr[B]$, we can rewrite

$$\begin{aligned}
P' &:= Pr[h(L) \in H_p \mid M_f(L) = M_f(l), k(L) = k(l)] \\
&= \frac{Pr[h(L) \in H_p, M_f(L) = M_f(l), k(L) = k(l)]}{Pr[M_f(L) = M_f(l), k(L) = k(l)]} \\
&= \frac{\sum_{l' : h(l') \in H_p, k(l) = k(l')} Pr[M_f(l') = M_f(l)] \cdot Pr[L = l']}{\sum_{l' : k(l) = k(l')} Pr[M_f(l') = M_f(l)] \cdot Pr[L = l']} \\
&= \frac{1}{1 + \frac{\sum_{l' :\ h(l') \notin h(l), k(l') = k(l)} Pr[M_f(l') = M_f(l)] \cdot Pr[L = l']}{\sum_{l'' :\ h(l'') \in h(l), k(l'') = k(l)} Pr[M_f(l'') = M_f(l)] \cdot Pr[L = l'']}} \quad ,
\end{aligned}$$

For an ε-DP mechanism M_f, since l' and l'' differ in one item due to the condition $k(l') = k(l) = k(l'')$, we have $\frac{Pr[M_f(l') = M_f(l)]}{Pr[M_f(l'') = M_f(g)]} \geq exp(-\varepsilon \cdot r)$, where r is the largest possible difference between two values of an attribute that the attacker is

guessing. This gives us

$$\begin{aligned} P' &\leq \frac{1}{1+exp(-\varepsilon \cdot r)\frac{\sum_{l':\ h(l')\notin h(l),k(l')=k(l)} Pr[L=l']}{\sum_{l'':\ h(l'')\in h(l),k(l'')=k(l)} Pr[L=l']}} \\ &= \frac{1}{1+exp(-\varepsilon \cdot r)\frac{Pr[h(L)\notin H_p,k(L)=k(l)]}{Pr[h(L)\in H_p,k(L)=k(l)]}} \\ &= \frac{1}{1+exp(-\varepsilon \cdot r)\frac{Pr[h(L)\notin H_p \mid k(L)=k(l)]}{Pr[h(L)\in H_p \mid k(L)=k(l)]}} \end{aligned}$$

Substituting P from Def 4.1.1, we get

$$P' \leq \frac{1}{1+exp(-\varepsilon \cdot r)\frac{1-P}{P}} \quad . \tag{4.3}$$

■

Laud et al. [84] quantify the maximum value of ε, that achieves the upper bound δ.

Proposition 4.2.2. *The maximum possible ε (i.e., the minimum noise) that achieves the upper bound δ, w.r.t. the above attack model, and minimizes the difference between the original and anonymized timestamp (c.f. UR2) is*

$$\varepsilon_k = -\ln\left(\frac{P_k}{1-P_k}\left(\frac{1}{\delta+P_k}-1\right)\right)\cdot\frac{1}{r} \quad , \tag{4.4}$$

where r is the maximum value in the range of values ($r=1$ after the normalization of values), and P_k is the prior guessing probability for an instance k.

Proof. Given Def 4.2.9, δ is the maximum guessing advantage probability after disclosing the attribute. Therefore,

$$P' - P \leq \delta \quad . \tag{4.5}$$

Substituting Eq (4.3) into Eq (4.5), we can estimate the largest possible ε (the minimum amount of noise) that achieves the upper bound δ as

$$\varepsilon = \frac{-\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right)}{r} \quad . \tag{4.6}$$

From Eq (4.6), the ε represents the maximum possible value for ε achieving the bound δ. Hence, we use Eq (4.6) to get the minimum possible noise that minimizes the difference between the original and anonymized timestamp. ■

Given Prop. 4.2.2, if there is enough data, we can estimate P_k based on the data distribution as:

$$P_k = CDF(t_k + p \cdot r) - CDF(t_k - p \cdot r) \quad , \tag{4.7}$$

where t_k is the value of an instance, and p is the precision. Suppose we cannot estimate the probability distribution of input values (e.g., the likelihood of participating in a subtrace). In that case, we can compute the worst-case scenario P_k as [84]:

$$P_k = (1 - \delta)/2 \text{ for all } k \quad . \tag{4.8}$$

4.3. Computing Differentially Private Event Logs

Given the above definitions, this section translates the guessing advantage parameter δ, into an algorithm that anonymizes the event log. Fig. 5 outlines the proposed approach. First, we construct the DAFSA state annotated event log as mentioned in Sect. 4.2.1. We use the algorithm proposed by Reissner et al. [81] to build the DAFSA. Second, we calculate the ε values needed by the approach. Finally, we use the calculated ε values to perform oversampling of cases and time noise injection into the event log. Below, we describe the steps in detail.

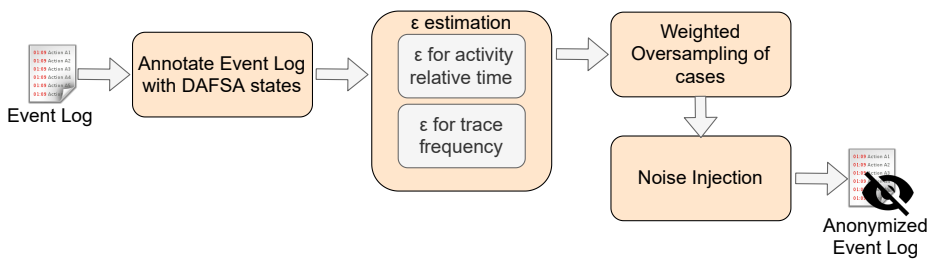


Figure 5. Approach

4.3.1. ε Estimation

Given the DAFSA annotated event log, we group the common prefixes and suffixes in traces by the DAFSA transitions. This grouping has two outputs: a distribution of execution time of activity instances that goes through the same DAFSA transitions and the DAFSA transitions contingency table. A case's time attribute comprises two components: the start time of the case and the relative execution time of every activity in the case (a time difference between its execution timestamp and the start timestamp of the case) in a time unit.

To calculate ε for the distribution of execution time instances, grouped by a DAFSA, we use Eq (4.7) and (4.4), i.e., we use different ε values for every event in the event log. We normalize the input values to become in the range $[0, 1]$ as mentioned in the above section. Eq (4.4) provides the maximum ε to address UR2 (the difference between the real and the anonymized time values is minimal).

On the other hand, the DAFSA transition contingency table contains a count histogram of the common prefixes and suffixes. To calculate the ε value for the marginals, we use Eq (4.8) and (4.4). The ε of all the DAFSA transitions of the contingency table is the same, but the noise is drawn for every transition independently. The propositions and the privacy proofs of ε estimation are presented in the supplementary material [86].

4.3.2. Weighted Oversampling of Cases

Given the above-calculated ε for trace frequency, we draw a random noise value from the Laplace distribution $Lap(\Delta f/\varepsilon)$, for each DAFSA transition. Adding or removing a prefix/suffix of a trace to the log affects a single frequency count by 1, so $\Delta f = 1$. We take the absolute value of the noise (additive noise) to fulfill **UR1a** The noise (quantified by δ) is added as increments to the frequencies of the contingency table. Using an absolute value (additive noise only), we alter the privacy guarantees of the differentially private mechanism, as the attacker can break Def. 2.4.1 if the sampled noise falls within $[0, \Delta f]$. This happens with probability $1 - exp(-\varepsilon/\Delta f)$. This additional risk is factored into δ.

Next, we increment the counts of the contingency table by means of oversampling of DAFSA transitions as defined in Def. 4.2.7. Finally, to maintain the consistency of the DAFSA transitions, we oversample the traces in the log using Def. 4.2.7, as presented in Algorithm 1. This algorithm does not suppress nor add case variants to fulfill **UR1a** . Sect 4.3.4 provides the correctness proofs of Algorithm 1.

Algorithm 1: Case Oversampling Algorithm

```
1.  Input: Event Log, ε, DAFSA
2.  Output: Differentially Private Event Log
3.  DafsaLookup = Build DAFSA transitions to case variant lookup ;
4.  DafsaLookup[i].neededNoise= |z_i|, where z_i is sampled from Lap(Δf/ε) independently for
      every transition t_i;
5.  cnt = count(DafsaLookup .addedNoise < DafsaLookup.neededNoise) ;
6.  while cnt > 0 do
7.      selectedTransition = pick a random transition such that DafsaLookup.addedNoise <
          DafsaLookup.neededNoise;
8.      pickedTraces = pick x random traces that traverse selectedTransition, where x =
          selectedTransition.neededNoise;
9.      foreach t ∈ pickedTraces do
10.         DafsaLookup[t].addedNoise ++ ;
11.         replicate a random case with a case variant = t ;
12.     end
13.     cnt = count(DafsaLookup.addedNoise < DafsaLookup.neededNoise);
14. end
15. generate new CaseID for every case;
16. shuffle cases ;
```

Algorithm 1 starts by constructing a correspondence (lookup) table between the DAFSA transitions and the case variants (line 3). This table maps every DAFSA

transition to the case variants that traverse it. We use this lookup table to track the updates over transitions. Second, we draw a random noise from the Laplace distribution $Lap(\Delta f/\varepsilon)$ (line 4) independently for every transition. Next, we count the DAFSA transitions that need noise injection and their needed noise (lines 5 and 6). Next, we pick a random transition that needs noise (with sampling weights of their occurrence frequency) (line 7). Then, we randomly choose a case variant that goes through the chosen transition (with sampling weights of their number of instances) (line 8). We replicate the chosen case variant by a number of times equals the needed noise (lines 9-11). For every replication, we choose a random case variant instance from the log to replicate. Next, we update the DAFSA lookup with the injected noise (line 10). We repeat this process until all the transitions have the minimum required noise. We keep ε values of the time attribute the same as the original cases to draw noise, as in Sect. 4.3.3. Then, we generate new case IDs for the cases (line 15), and we change their order (line 16).

4.3.3. Noise Injection

At this step, we have the DAFSA annotated event log, with the oversampled case instances and an ε value of each event. First, we divide the ε value of the replicated cases by the number of replications, as oversampling is considered repeating the same query more than once [6]. Then, we draw a random noise from the Laplace distribution $Lap(\Delta f/\varepsilon)$ to anonymize the relative time for every activity instance. Next, we transform the amount of noise from the normalized range ($[0,1]$) to the original range. Finally, we add the time noise to the original execution relative time. After the noise injection, we transform the relative execution time of activities to timestamps. By the end of this step, the event log is anonymized by ε values calculated for the input maximum guessing advantage δ.

4.3.4. Correctness proofs of Algorithm 1

In Def 4.2.5, we defined the DAFSA transition Contingency table as a histogram of counts of DAFSA transitions. First, we estimate the privacy leakage through the contingency table.

Proposition 4.3.1. *Let M^{CT} be a mechanism that, for each count cell, samples (independently) noise from the Laplace distribution $Lap(1/\varepsilon)$ and adds it to the count. Then, the level of DP w.r.t. change in a prefix/suffix of some trace of the underlying event log is:*

1. *ε for a single count cell of the noisified CT;*
2. *$k \cdot \varepsilon$ for the entire noisified CT, where k is the longest case length in the event log.*

Proof. Suppose that we update a prefix/suffix of a trace in the event log. Each single count cell in the contingency table may change by at most ± 1 for each update step. Hence, the global sensitivity of a single count cell w.r.t. changing

some prefix/suffix is 1. We can sample additive noise from distribution $Lap(1/\varepsilon)$ and add it to a count cell to achieve ε-DP w.r.t. that count cell.

For the entire contingency table, a change in the frequency of a single case variant can affect at most k count cells (i.e., the count over the DAFSA transitions that represent that case variant), where k is the maximum case variant length in the event log. Hence, the global sensitivity is k, and we need to sample noise from $Lap(k/\varepsilon)$ to achieve ε-DP. ■

Given the desired guessing advantage threshold δ, we need to prevent an attacker from achieving the attack h_1.

Proposition 4.3.2. *Let M^{CT} be a mechanism that, for each count cell, draws (independently) noise from the Laplace distribution $Lap(1/\varepsilon)$, adds or subtracts the noise from the count, and rounds negative results to 0. Let δ be the desired upper bound probability that the attacker achieves attack h_1. Then, it suffices to take*

$$\varepsilon = -\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right) \ ,$$

where $P = \dfrac{1-\delta}{2}$.

Proof. Let P be the prior probability of guessing (i.e., without observing the output). We adopt the mechanism proposed by Laud et al. [84, 85] to estimate the guessing advantage, which is the upper bound of the difference between the posterior and the prior probability of guessing an attribute ranging between 0 and 1 (r=1) by

$$\delta = \frac{1}{1+exp(-\varepsilon)\frac{1-P}{P}} - P \ ,$$

which can be reversed to

$$\varepsilon = -\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right) \ .$$

Laud et al. [84,85] estimate the prior knowledge P from the distribution of input values. However, the contingency table contains counts, which means the lack of distribution. Laud et al. [84, 85] elaborate that, in case of the lack of distribution, we can estimate the value of P that minimizes the ε. We take the derivative of ε w.r.t. P, which yields $P = \dfrac{1-\delta}{2}$ as the prior knowledge for the contingency table counts.

We need to sample noise to achieve ε-DP. First of all, rounding negative noisified counts up to 0 can be considered as post-processing that does not depend on private data. Hence, we focus on adding the Laplace noise. While noise sampled from $Lap(1/\varepsilon)$ would be enough for a single count cell, for an entire table, we would need $Lap(k/\varepsilon)$ where k is the longest case length in the event log, as shown in Prop 4.3.1.

All correlated cells may provide additional information about the target cell. For example, activity B may always follow activity A. This additional knowledge may increase P. However, we have already chosen the *worst-case P* (in terms of guessing advantage) for estimating the noise, which does not depend on any background knowledge that the attacker may get, including the related outputs. We note that if we used another prior knowledge estimation and/or estimated the attacker's success directly instead of estimating the *advantage*, then we would indeed require sampling noise from $Lap(k/\varepsilon)$. ■

Correctness of Oversampling. In this chapter, we address RQ1 under the requirement that "the anonymized log must have the same set of case variants as the original log." We use *oversampling* to keep all the initial entries in the event log.

Proposition 4.3.3. *Let the event log L have a fixed timestamp value for all the entries. Let CT be the contingency table that corresponds to the event log L. Let M^{OV} be a mechanism that, for each count cell of the CT, samples (independently) noise z from Laplace distribution $Lap(1/\varepsilon)$ and:*

- *Inserts into L $|z|$ additional copies of any cases that have been included in that count cell;*
- *Shuffles all resulting cases and updates all case IDs before publishing the resulting event log.*

Let δ be the desired upper bound of the attacker succeeding in the goal h_1. Then,

- *it suffices to take*

$$\varepsilon = -2\ln(b/c^2 - (\delta-1)/(c\cdot b)) \ ,$$

where $c = \sqrt[3]{6}$ and

$$b = \sqrt[3]{\sqrt{3}\cdot\sqrt{2\cdot\delta^3+21\cdot\delta^2-48\cdot\delta+25}-9\cdot\delta+9}.$$

- *a mechanism that adds noise $|z|$, where $z \leftarrow Lap(1/\varepsilon)$, fulfills the requirement that "the anonymized log must have the same set of case variants as the original log."*

Proof. Oversampling the cases using Def 4.2.7 keeps the same case variants as the input log L. We show that a mechanism M that adds noise $|z|$ for $z \leftarrow Lap(1/\varepsilon)$ to the counts of CT constructed from L ensures the differential privacy guarantees. Let L_1 and L_2 be two neighboring event logs that differ in the presence of one trace. Without loss of generality, let L_1 have a smaller count than L_2 for the observed count cell. Suppose that the added noise instance is $|z| \geq 1$. In that case, the noisified output y can be obtained from both L_1 and L_2. In particular, we have

$$\begin{aligned}\frac{\Pr[M(L_1)=y]}{\Pr[M(L_2)=y]} &= \frac{exp(\varepsilon\cdot(y-M(L_1)))}{exp(\varepsilon\cdot(y-M(L_2)))} \\ &= exp(\varepsilon\cdot(y-M(L_1)+y-M(L_2))) \\ &\leq exp(\varepsilon) \ .\end{aligned}$$

So the guessing advantage δ for $|z| \geq 1$ can be computed as in Prop 4.3.2. However, for $|z| < 1$, the noisified value y produced by $M(L_1)$ will be *between* the true counts of L_1 and L_2, and y could never be an output of $M(L_2)$, since we only add positive noise. The probability of getting $|z| \geq 1$ for Laplace noise is

$$CDF_{LAP}(1) - CDF_{LAP}(-1) = (1 - \frac{1}{2}exp(-\varepsilon)) - \frac{1}{2}exp(-\varepsilon) = 1 - exp(-\varepsilon).$$

Let ε be computed to achieve guessing advantage δ' with standard Laplace distribution, as in Prop 4.3.2. The actual guessing advantage with one-sided Laplace distribution will be

$$\delta = exp(-\varepsilon) \cdot \delta' + (1 - exp(-\varepsilon)) \quad . \tag{4.9}$$

From Prop 4.2.2, the maximum ε that achieves the upper bound δ is

$$\varepsilon = -\ln\left(\frac{P}{1-P} \cdot \left(\frac{1}{\delta' + P} - 1\right)\right),$$

for $P = \dfrac{1-\delta'}{2}$. Substituting from (4.6) into (4.9) and using a numerical solver (e.g., Wolfram Alpha) to express ε through δ.

$$\varepsilon = -2\ln(b/c^2 - (\delta - 1)/(c \cdot b)) \quad ,$$

where $c = \sqrt[3]{6}$ and

$$b = \sqrt[3]{\sqrt{3} \cdot \sqrt{2 \cdot \delta^3 + 21 \cdot \delta^2 - 48 \cdot \delta + 25} - 9 \cdot \delta + 9}.$$

■

Timestamp Anonymization. In this appendix, we consider timestamp anonymization. We need to add enough noise to the timestamps to prevent attacker success in h_2, and h_1, since timestamps may potentially leak which traces are real and which are not (new injected cases by sampling). We note that correlations between timestamps are difficult to handle in terms of guessing advantage. The main problem is that the effect of correlated times would greatly depend on how they are correlated, and in some cases, even a little leakage of t_2 may leak everything about t_1. For example, let $t_1 \in \{0,1\}$ and $t_2 \in \{0,...,1023\}$ be distributed uniformly. We have the prior knowledge $P_1 = 1/2$, and $P_2 = 1/1024$. Suppose that, after seeing the noisified output, the attacker constrained his view of t_2 to $\{0,...,511\}$. There are now 512 possible choices for t_2, so $\delta = 1/512 - 1/1024 = 1/1024$, which is very small. However, when t_1 is the highest bit of t_2, the value of t_1 would be leaked. This particular correlation of times is unlikely in practice, but it demonstrates the problem in general.

To this end, we convert a timestamp t_k to a time difference $dt_k := t_k - t_{k-1}$ of sequential events, which is the duration of time that an individual has spent in a

transition from one event to the next one. The timestamp at time t_0 is the starting time st, which needs to be published as well to make it possible to reconstruct the actual timestamps. st can be converted to the time difference between the start time of a case and the start time of the log. We obtain the result for linearly correlated time differences (e.g., increasing the duration of the first event by a minute increases every other event at most by one minute). To this end, we scale ε_k by the length of the trace. Prop 4.3.4 proposes a mechanism that anonymizes the time differences.

Proposition 4.3.4. *Let M^T be a mechanism that, for each timestamp t_k, samples (independently) noise from the Laplace distribution $Lap(1/\varepsilon)$ and adds the noise to t_k. Let r_k the maximum possible value of t_k. Let δ be the desired upper bound of the attacker succeeding in the attack h_2 with precision p. Then, it suffices to take*

$$\varepsilon_k = -\frac{\ln\left(\frac{P_k}{1-P_k}\cdot\left(\frac{1}{\delta+P_k}-1\right)\right)}{m\cdot r_k} \quad ,$$

where $P_k = CDF(t_k + p\cdot r_k) - CDF(t_k - p\cdot r_k)$, CDF is the probability density function of the distribution of times, and m is the length of the longest trace in L.

Proof. We can estimate an upper bound on the difference between the posterior and the prior probability of guessing an attribute ranging between 0 and 1 by,

$$\delta = \frac{1}{1+exp(-\varepsilon_k\cdot r_k)\frac{1-P_k}{P_k}} - P_k \quad ,$$

which can be reversed to

$$\varepsilon_k = -\frac{\ln\left(\frac{P_k}{1-P_k}\cdot\left(\frac{1}{\delta+P_k}-1\right)\right)}{r_k} \quad .$$

The quantity P_k is defined as the probability of a value being in the interval $[t_k - p\cdot r_k, t_k + p\cdot r_k]$. From Prop 4.2.2, the above value of ε is the maximum within the upper bound δ. If CDF of time distributions is unknown, it is safe to take P_k that minimizes the ε_k (and hence maximizes the amount of noise), which is $P_k = \frac{1-\delta}{2}$ for all k.

That was about leakage for a single published timestamp. We could keep ε_k the same for an entire L if different times are not correlated. If the times are linearly correlated, then, by the sequential composition of DP, we need to divide ε_k by the length of the longest trace m. ■

Event Log Anonymization. We now provide a proof of correctness of Alg 1.

Lemma 4.3.5. *Let L and δ be an event log and a maximum guessing advantage threshold. Log $L' = Alg1(L,\delta)$ fulfills the two properties in Def 4.1.1 (i.e., L' is ε-differentially private), and L' fulfills requirement **UR1a**, and the difference between the real and the anonymized time values is minimal (c.f. UR2).*

Proof. We first show that L' fulfills the two properties in Def 4.1.1

(1) In Prop. 4.3.3, we proved that it suffices to take

$$\varepsilon_d = -\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right)$$

to anonymize the case variants of an event log in order to provide differential privacy guarantees for the desired guessing advantage upper bound δ. Algorithm 1 annotates the event log using DAFSA transitions (line 3), and computes the contingency table CT of the log. After that, the algorithm uses the above ε_d to oversample cases (replication cases) (lines 9 and 11). Given that Prop. 4.3.2 and Prop. 4.3.3 prove that replicating cases with a random sample size z from Laplace distribution $Lap(1/\varepsilon)$ provides differential privacy guarantees. Algorithm 1 provides differential privacy guarantees for the case variants anonymization;

(2) In Prop. 4.3.4, we proved that it suffices to take

$$\varepsilon_k = -\frac{\ln\left(\frac{P_k}{1-P_k}\cdot\left(\frac{1}{\delta+P_k}-1\right)\right)}{r_k}$$

to provide differential privacy guarantees w.r.t timestamp for the desired guessing advantage upper bound δ. Given that Algorithm 1 adopts the above ε to anonymize timestamp each event (line 27). Hence, Algorithm 1 provides differential privacy guarantees for the event log anonymization w.r.t. timestamp.

We now show that L' fulfills the requirement **UR1a** with minimal noise injection for timestamp(cf. **UR2**).

1. Algorithm 1 does not create new case variants. It only replicates existing variants (cf. lines 9 and 11). Hence, it fulfills requirement **UR1a**;
2. In Prop. 4.2.2, we proved that the ε_k as defined in Eq (4.6) is the maximum within the upper bound δ, so Algorithm 1 injects minimal noise injection for timestamp.

■

4.4. Summary

This chapter proposed a concept of differentially private event log and a mechanism to compute such logs. A differentially private event log limits the increase in the probability that an attacker may learn a suffix of an individual's trace given a prefix (or vice-versa), or the timestamp of an activity in an individual's trace. To this end, we inject differentially private noise by oversampling the traces in the log. This approach neither suppresses nor adds case variants (cf. Sect. 5.1.5) and hence fulfills **UR1a**. To address UR2 (the difference between the real and the

anonymized time values is minimal), we quantify ε based on a technique that finds the maximum ε (minimum noise) that keeps the guessing advantage below δ (cf. Sect.4.2.1 and Prop. 4.2.2).

The proposed method introduces high levels of noise in the presence of unique traces or temporal outliers. To address this limitation, we plan to investigate an approach where high-risk traces are suppressed so that the amount of injected noise into the remaining traces is lower. Although suppression can significantly reduce the required noise level and strengthen the privacy guarantees, it breaks the property that the differentially private log has the same case variants as the original one. To mitigate this drawback, Chapter 5 seeks to define an approach to minimize the number of case variants that need to be suppressed, given the desired level of guessing advantage.

5. DIFFERENTIALLY PRIVATE RELEASE OF EVENT LOGS: AN UNDER- AND OVERSAMPLING APPROACH

The differentially-private approach presented in Chapter 4 injects high level of noise in event logs with unique traces or temporal outliers. We observe that suppressing the high-risk traces reduces the amount of required noise to be injected. This chapter addresses RQ1 under the utility requirement *UR1b: The anonymized event log must not introduce new case variants to the original log.* and the utility requirement *UR2: The difference between the real and the anonymized time values is minimal given a privacy risk metric.*

This chapter extends the work in Chapter 4 under UR1b. We propose a revised anonymization method, which achieves lower utility loss for a given guessing advantage δ by: (i) applying both over- and undersampling, as opposed to only oversampling; and (ii) filtering out high-risk cases. This chapter also presents an experimental setup to assess utility loss not only w.r.t. the distance between the anonymized and the original log, but also w.r.t. the impact of anonymization on the process maps discovered from the (anonymized) log.

The chapter is structured as follows. Sect. 5.1 presents the proposed approach to anonymize event logs. Sect. 5.2 presents an open source tool for event log anonymization. Sect. 5.3 presents an empirical evaluation. Finally, Sect. 5.4 concludes the work. This chapter is derived from [87] and contains sentences or fragments of sentences from this prior publication.

5.1. Approach

We seek to anonymize an event log in such a way that an attacker cannot single out an individual based on a prefix or suffix of the individual's trace or based on the event timestamps. Accordingly, the proposed approach relies on a data structure that captures all prefixes and suffixes of a set of traces, namely a Deterministic Acyclic Finite State Automata (DAFSA) [81]. By analyzing the frequency and time differences of each DAFSA transition, we estimate the amount of noise required to achieve the required guessing advantage.

Concretely, given a log and a guessing advantage threshold δ, our approach produces a differentially private event log in 7 steps as outlined in Fig. 6.

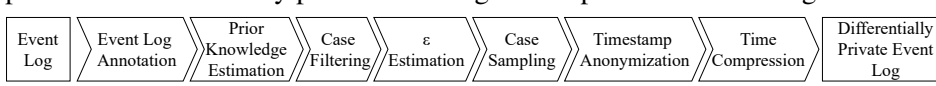


Figure 6. Approach

The first step constructs a lossless intermediate representation of a log (DAFSA). The DAFSA groups the traces that share the same prefixes or suffixes. An attacker may have prior knowledge about the events recorded in the log before releasing

the log, e.g., in the hospital log in Table 1, the attacker has a prior knowledge that probability that a blood test activity has been executed for Lena is 20%. To address this issue, we estimate the prior knowledge of every event that belongs to a prefix/suffix group. Some events may correlate with high prior-knowledge values, leading to more noise injected in the log to achieve the given δ threshold. Accordingly, we provide a case filtering mechanism that filters out entire cases based on the estimated prior knowledge of their events. The fourth step estimates an ε value for every prefix/suffix group for the input δ threshold. The fifth step uses the estimated ε value to apply sampling to the cases based on their prefixes/suffixes and case variants. The sixth step applies timestamp anonymization based on the estimated ε values for every group of prefixes/suffixes. Lastly, we post-process the differentially private log to compress the timestamp values so that the overall timeframe of the resulting log matches closely with the original log. We generate new case IDs, so an attacker cannot use the case ID (on its own) to identify an individual.

This chapter adopts the attack model presented in Sect 4.1. The rest of this section discusses each step of the proposed approach in turn.

5.1.1. Event Log State-Annotation

Our goal is to prevent an attacker from singling out individuals based on any prefix or suffix of their trace (cf. attack goal h_1). To this end, we group the prefixes and suffixes in the log and inject independent differentially private noise to each group. Consequently, we use the DAFSA event log representation presented in Sect. 4.2.1.

The DAFSA-annotated event log links every event in the log, based on its prefix, suffix, and activity label, to a DAFSA transition. Every event is labeled by the source state and target state of the DAFSA transition. Table 7 (columns "Src. state" and "Tgt. state") shows the DAFSA-annotated log.[1]

Timestamps represent the time at which every event happened. However, it is more beneficial to compare the duration of activities in every group of suffixes and prefixes to prevent singling out an individual based on the duration of their activity and its timestamp. We calculate the relative time to compare the time within a group of traces that share the same suffix/prefix and activity label. The relative time of an event is the time difference between an event and its successor. Table 7 (column Rel. Time) shows the relative time estimated for the given events. For the start event of every case, the relative time is the difference between the case start event and the first event in the event log.

5.1.2. Prior Knowledge Estimation

Without publishing the log, attackers can use their knowledge to guess information about a specific individual. We estimate the prior knowledge of an attacker using the framework proposed in [84]. An attacker's guess $h(L)$ is considered successful

[1] Columns "Norm. Rel. Time", "Prec." and "PK" are explained later.

Table 7. DAFSA State-Annotated Event log

Case ID	Act. Label	Timestamp	Src. State	Tgt State	Rel. Time	*Nrm. Rel. Time*	*Prec.*	*PK*
1	A	8/8/2020 10:20	s_0	s_5	0	0	1	0.33
	B	8/8/2020 10:50	s_5	s_2	30	0.33	0.67	0.75
	C	8/8/2020 16:15	s_2	s_3	325	0.0	0.01	0.33
2	D	8/8/2020 12:37	s_0	s_4	0	0	1	0.33
	A	8/8/2020 14:37	s_4	s_5	120	1	0	0.35
	E	8/8/2020 15:07	s_5	s_2	30	1	0	0.35
	C	8/8/2020 20:31	s_2	s_3	324	0	0.01	0.33
3	A	8/9/2020 13:30	s_0	s_5	0.99	0.33	1	0.5
	B	8/9/2020 13:55	s_5	s_2	25	0	0.67	0.5
	C	8/9/2020 20:55	s_2	s_3	420	0.07	0.01	0.167
4	D	8/9/2020 15:00	s_0	s_4	0.99	1	1	0.5
	A	8/9/2020 17:00	s_4	s_5	120	1	0	0.35
	B	8/9/2020 17:40	s_5	s_2	40	1	0.67	0.25
	C	8/9/2020 23:05	s_2	s_3	325	0.0	0.01	0.33
5	A	8/9/2020 17:25	s_0	s_5	0.99	0.33	1	0.5
	E	8/9/2020 17:55	s_5	s_2	30	1	0	0.35
	C	8/10/2020 23:55	s_2	s_3	1800	1	0.01	0.17
6	A	8/11/2020 17:00	s_0	s_5	3	1	1	0.17
	B	8/11/2020 17:27	s_5	s_2	27	0.13	0.67	0.5
	C	8/11/2020 23:45	s_2	s_3	378	0.04	0.01	0.17

if it falls within a range of values H_p, which is the actual value $\pm$ precision (c.f. Def 4.2.9).

A guessing precision p is a percentage value representing the range of a successful guess H_p. For example, if the time between two executive events of the same trace in a log is 0.5 hour, and $p = 0.2$ (12 minutes), the guessed value is considered successful if it falls in range $H_p = [0.3..0.7]$ hour, i.e., [18 .. 42] minutes. To interpret the precision in the range of values [0,1] (a percentage value), we pre-process the log to normalize the range of values (relative timestamps) to be in the range [0,1]. We assume that the precision for the start timestamp is one day and the precision for the relative time is 10 seconds. Table 7 (columns Nrm. Rel. Time and Prec.) shows the normalized relative time (the normalization is based on the event's DAFSA transition group) and the estimated precision.

To apply the prior knowledge to the DAFSA-annotated log, we group the events based on their DAFSA transition (source state, activity, and target state). For each group of values, we estimate the prior knowledge using Eq. 4.7. Table 7 (column PK) shows the estimated prior knowledge for every instance with $\delta = 0.3$.

5.1.3. Case Filtering

In some cases, the prior knowledge P_k is very high such that more noise is needed to keep the guessing advantage below the threshold δ after publishing the event log. We filter out the instances that violate $P_k + \delta \geq 1$ to reduce the injected noise in such a case. To fulfill **UR1b**, we filter out the entire case.

Definition 5.1.1 (Case Filtering)**.** A case filter F is a function that filters out cases with at least one DAFSA transition that violates the condition $P_k + \delta \geq 1$.

In Table 7, activity B of the first case has a prior knowledge value $P_k = 0.75$ and $\delta = 0.3$. Our approach filters out the entire first case because at least one DAFSA transition violates the condition $P_k + \delta \geq 1$. Table 8 shows the filtered DAFSA-annotated event log.[2] After case filtering, we re-estimate the prior knowledge for each event. Table 8 (column New PK) presents the newly estimated prior knowledge values. Case filtering does not have an impact on privacy because it does not depend on private data.

5.1.4. ε Estimation

Given the filtered DAFSA-annotated event log, we need to estimate the amount of noise in order to anonymize the event log. We use DP, which quantifies the noise using ε. One naive approach is using the same ε both for control-flow and timestamp anonymization. The two types of queries are different, and the quantity of noise has a different impact on each. Given the attack model in Sect. 4.1, for attack h_1, we use the lossless log representation (DAFSA) to apply an ε-DP mechanism to the prefixes and suffixes of the traces in the log. For attack h_2,

[2] The column "ε_t" is explained later.

Table 8. Filtered DAFSA State-Annotated Event log

Case ID	Act. Label	Timestamp	Src State	Tgt State	Rel. Time	New PK	ε_t
2	D	8/8/2020 12:37	s_0	s_4	0	0.2	1.39
	A	8/8/2020 14:37	s_4	s_5	120	0.35	1.24
	E	8/8/2020 15:07	s_5	s_2	30	0.35	1.24
	C	8/8/2020 20:31	s_2	s_3	324	0.2	1.39
3	A	8/9/2020 13:30	s_0	s_5	0.99	0.6	1.8
	B	8/9/2020 13:55	s_5	s_2	25	0.33	1.24
	C	8/9/2020 20:55	s_2	s_3	420	0.2	1.39
4	D	8/9/2020 15:00	s_0	s_4	0.99	0.6	1.79
	A	8/9/2020 17:00	s_4	s_5	120	0.35	1.24
	B	8/9/2020 17:40	s_5	s_2	40	0.33	1.24
	C	8/9/2020 23:05	s_2	s_3	325	0.2	1.39
5	A	8/9/2020 17:25	s_0	s_5	0.99	0.6	1.79
	E	8/9/2020 17:55	s_5	s_2	30	0.35	1.24
	C	8/10/2020 23:55	s_2	s_3	1800	0.2	1.39
6	A	8/11/2020 17:00	s_0	s_5	3	0.2	1.39
	B	8/11/2020 17:27	s_5	s_2	27	0.33	1.24
	C	8/11/2020 23:45	s_2	s_3	378	0.2	1.39

we apply a bounded ε-DP mechanism w.r.t. the timestamp attribute (Def. 2.4.3). Consequently, we seek to quantify two ε values: the ε_d is required to anonymize the DAFSA, and the ε_t is required to anonymize the timestamp attribute.

Both ε_d and ε_t should guarantee that the attacker cannot single out an individual based on any prefix or suffix of their trace. Accordingly, we group the prefixes and suffixes in the log and estimate ε_d and ε_t for each group.

Estimating ε_t. To protect against attack goal h_2, we need to inject noise sufficient to achieve a given ε_t, determined by the guessing advantage threshold. Two approaches are possible. The first one is to calculate a single ε_t for every event in the log. The second approach is to estimate an epsilon that minimizes the noise injected into each event individually. We use the second one.

We estimate ε_t for an event based on the group of prefix that ends with the event and the suffixes that start with the event. Consequently, we adopt a *personalized differential privacy* mechanism that uses a different ε_t value for every event. Personalized DP assumes that every individual has independent privacy specification Φ [88]. In other words, $\Phi = (u_1, \varepsilon_1), (u_1, \varepsilon_1), ..., (u_m, \varepsilon_m))$, where an individual $u \in U$, and the event log has m users.

Definition 5.1.2 (Personalized Differential Privacy [88])**.** Given a universe of cases U, a mechanism M is said to be Φ-personalized differentially private if all the event logs L_1 and L_2 differing at most on one value of a tuple t, and all $S \subseteq Range(M)$.

$$Pr[M(L_1) \in S] \leq exp(\Phi^u) \times Pr[M(L_2) \in S]$$

where $u \in U$ is the case that corresponds to tuple t, and Φ^u denote u's privacy quantification.

We assume that the log publisher provides the maximum acceptable increase in the successful guessing probability after publishing the anonymized event log, called Guessing Advantage δ. Laud et al. [84] provide a framework that quantifies the ε value from a given guessing advantage δ. A formal definition of the guessing advantage is presented in Def. 4.2.10. Furthermore, Laud et al. [84] quantify the maximum ε value that achieves the upper bound δ for every data instance (c.f. Prop. 4.2.2).

To quantify ε using Eq (4.4), we need to study the distribution of values among every group of prefixes/suffixes. We quantify ε_t using the DAFSA-annotated event log to mitigate singling out an individual by their prefixes (or suffixes). Given Definitions 4.2.1, and 4.2.4, all cases that share a common prefix will traverse a given state s in the DAFSA corresponding to this prefix. The same holds for cases that share a common suffix.

To anonymize the timestamps, we anonymize two components: (1) the start time of the case, which we define as the time difference between the start time of the case and the first start time in the event log); and (2) the execution time of each activity in the case, defined as the difference between its execution timestamp and the timestamp of the successor activity in the same case.

To anonymize the start time of each case, we group all the start times of the cases in the log to anonymize the fact that a given case happened on a specific day. Accordingly, we group events that have source state s_0 as a single group. To anonymize the execution time of activities, we group the events that have the same source state, activity label, and target state in the DAFSA. For each group of the above, we use Eq (4.7) and (4.4) to estimate a different ε_t value for every event. Eq (4.4) provides the maximum ε_t to minimize the difference between the original and anonymized timestamp (c.f. UR2). The ε_t estimated for each event are shown in Table 8 (column ε_t).

Estimating ε_d. Parameter ε_d determines the noise to be applied to the occurrence count of each case variant in the log to prevent attack h_1. To estimate ε_d for a given group of prefixes or suffixes, we consider each group's size (count). This count is given by the following SQL query:

```
CT = SELECT SourceState, ActivityLabel, TargetState, COUNT(*)
FROM StateAnnotatedEventLog
GROUP BY SourceState, ActivityLabel, TargetState
```

The output of the above query is the DAFSA transition contingency table (CT) (c.f. Def. 4.2.5). We use the CT to estimate ε_d by using Eq (4.8) and (4.4). The ε_d of all the DAFSA transitions of the CT is the same, but the noise is drawn for every transition independently.

Table 9 shows the DAFSA transitions CT. These counts are called marginals. The marginals contain the correlations counts of the common sets of prefixes and

suffixes of the DAFSA. We anonymize the marginals to prevent singling out that an individual has been through an activity, using the prefix and the suffix set of activities. Barak et al. [82] use an unbounded ε-DP mechanism (cf. Definition 2.4.1) for marginal anonymization. Likewise, we use unbounded ε-DP to anonymize the DAFSA transitions CT.

5.1.5. Case Sampling

Given the mechanism M_f of Def. 4.2.6, we need to translate the noise estimated by the CT to the log. Kifer et al. [83] study the anonymization of CTs. They present the notion of a *move* to define the anonymization of the marginals of CTs. A move is a process that adds or deletes a tuple from the CT. We calculate the number of moves by drawing a random noise using ε_d. Then, we translate the moves to a sampling of the cases that go through the same DAFSA transitions. Sampling may require zero or more moves.

Table 9. DAFSA Transitions Contingency Table, and the generated Random Noise every DAFSA Transition for δ=0.3 and estimated ε_d=1.238

Source State	Activity	Target State	Count	Noise
s_0	A	s_5	3	0
s_5	B	s_2	3	2
s_2	C	s_3	5	1
s_0	D	s_4	2	-3
s_4	A	s_5	2	0
s_5	E	s_2	2	0

Chapter 4 has a stricter utility requirement (c.f. **UR1a**). It fulfills the requirement that "the anonymized log must have the same set of case variants as the original log." We defined oversampling as increasing a count (positive move) in the CT by replicating a random tuple in the log to fulfill such a requirement (c.f. Def. 4.2.7).

This chapter considers a relaxation of the above requirement (cf. **UR1b**). We extend our approach to use sampling instead of oversampling, i.e., our approach performs both replication and deletion of cases from the log. We define *sampling* as an increase or a decrease of a count in the CT.

The randomly generated sample size (based on ε_d) can have a positive or a negative size. We draw a random noise value from the Laplace distribution $Lap(\Delta f/\varepsilon_d)$, for each DAFSA transition. Table 9 shows the random noise for every DAFSA transition with the estimated ε_d= 1.238. The positive size is translated to replicating a random tuple in the log. The negative size is translated to deleting a random tuple in the log. Adding or removing a prefix/suffix of a trace to the log affects a single frequency count by 1, so $\Delta f = 1$.

Definition 5.1.3 (DAFSA Transition Sampling)**.** Given a DAFSA transition contingency table C_i, a sample $Sample$ is a transformation that adds or deletes a DAFSA

transition instance from C_i, producing a contingency table $C_j = Sample(C_i)$, with an increase or a decrease of only one count cell by 1.

To avoid inserting new trace variants in the event log (cf. **UR1b**), we sample the prefix and the suffix of the sampled transition, i.e., we sample an entire case that goes through the sampled transition.

Definition 5.1.4 (Case Sample)**.** Given an event log L, a case sample $Sample_{c_i}$ is a transformation that either duplicates or deletes a case c_i of log L, that goes through a sampled DAFSA transition t, in such a way that it duplicates or deletes all the activities of c_i.

Timestamp Noise Injection At this step, we have the DAFSA annotated log, with the sampled case instances and an ε_t value of each event. First, if some cases have been replicated, we divide the ε_t value of the replicated cases by the number of replications, as it is considered repeating the same query more than once [6]. Then, we draw a random noise from the Laplace distribution $Lap(\Delta f/\varepsilon_t)$ to anonymize the relative time for every activity instance and the start time of each case. Finally, we transform the relative execution time of activities to timestamps.

Algorithm 2 presents the steps we perform in order to calculate the differentially private event log. This algorithm does not introduce new case variants so as to fulfill **UR1b**.

Algorithm 2 starts by performing event log state-annotation as described in Sect. 5.1.1. Then, we estimate the prior knowledge as described in Sect. 5.1.2. After that, we perform case filtering as presented in Sect. 5.1.3. Then, we estimate ε_t, and ε_d as described in Sect. 5.1.4. Then, we construct a correspondence (lookup) table between the DAFSA annotated event log and the case variants (line 7). This table maps every DAFSA transition to the case variants that traverse it. Also, we use this lookup table to track the updates over transitions. Second, we independently draw a random noise from the Laplace distribution $Lap(\Delta f/\varepsilon_d)$ (line 8) for every transition. We initialize added noise counter to be zero (line 9). Next, we count the DAFSA transitions that need noise injection and their needed noise (lines 10). Next, we choose a random transition that needs noise (with their occurrence frequency as sampling weights) (line 12). Then, we randomly choose a case variant that goes through the chosen transition (with sampling weights of their number of instances) (line 13). Next, based on the needed noise (line 15), we either replicate (lines 16-17) or delete (lines 20-21) the chosen case variant by a number of times equals the needed noise. For every replication/deletion, we choose a random case variant instance from the event log to be replicated. We repeat this process until all the transitions have the minimum required noise. Finally, we divide ε_t of the replicated cases by the number of replications (line 26). By the end of this step, the event log is anonymized by ε values calculated from the input maximum guessing advantage δ.

Next, we show that the output of Algorithm 2 indeed ensures the ε-differential privacy guarantees.

Algorithm 2: Event Log Anonymization Algorithm

Input: L: Event Log, δ: Guessing Advantage Threshold
Output: L': ε-Differentially Private Event Log
L= eventLogStateAnnotation(L);
p_k=priorKnowledgeEstimation(L,δ);
L_f= caseFiltering(L, p_k);
ε_d, ε_t= εEstimation(L_f,p_k, δ);
DafsaLookup = Build DAFSA annotated event log to case variant lookup(L_f) ;
DafsaLookup[i].neededNoise= z_i, where z_i is sampled from $Lap(\Delta f/\varepsilon_d)$ independently for every transition t_i;
DafsaLookup .addedNoise=0;
cnt = count(|DafsaLookup .addedNoise| < |DafsaLookup.neededNoise|) ;
$L' = L_f$;
while *cnt > 0* **do**
    selectedTransition = pick a random transition such that |DafsaLookup.addedNoise| < |DafsaLookup.neededNoise|;
    pickedTraces = pick x random traces that traverse selectedTransition, where x = selectedTransition.neededNoise;
    foreach $t \in$ *pickedTraces* **do**
        if *DafsaLookup[t].neededNoise > 0* **then**
            DafsaLookup[t].addedNoise ++ ;
            add to L' a replica of a random case with a case variant = t ;
        else
            DafsaLookup[t].addedNoise −− ;
            delete from L' a random case with a case variant = t ;
        end
    end
    cnt = count(|DafsaLookup .addedNoise| < |DafsaLookup.neededNoise|) ;
end
DafsaLookup[i].ε_{t_i}=DafsaLookup[i].ε_{t_i}/DafsaLookup[i].NumOfReplicas, where i is the replicated cases;
L'.timestamp= L'.timestamp + z_i, where z_i is sampled from $Lap(\Delta f/$DafsaLookup[i].$\varepsilon_{t_i})$
return L'

5.1.6. Privacy Proof of Algorithm 2

Correctness of Sampling. In this chapter, we adopt differential privacy, which injects noise into the data before its publication. In order to translate the calculated noise for every DAFSA transition group (in CT) into case variants anonymization, we perform case sampling, as defined in Def 5.1.3 and Def 5.1.4. Following, we provide the correctness of sampling. We assume that the event log contains the three columns: Case ID, Activity label, and Timestamps, and the activity labels are public information. First, we discuss privacy leakage of the case variants distribution (*without* taking into account timestamps).

Proposition 5.1.1. *Let the event log L have a fixed, constant timestamp value for all the entries. Let CT be the contingency table that corresponds to the event log L. Let M^{OV} be a mechanism that, for each count cell of the CT, draws (independently) noise z from Laplace distribution $Lap(1/\varepsilon)$ and:*

- *Replicates the random cases that go through the DAFSA transition, which corresponds to the count cell (prefix/suffix group), if $z > 0$. The number of replications is z;*
- *Drops random cases from the event log that go through a DAFSA transition which corresponds to the count cell, if $z < 0$. The number of deletions is z;*
- *Shuffles all cases and updates all case IDs before publishing the resulting event log.*

Let δ be the desired upper bound of the attacker succeeding in the goal h_1. Then, it suffices to take

$$\varepsilon = -\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right) \quad ,$$

where $P = \dfrac{1-\delta}{2}$.

Proof. We show that sampling the cases has the same privacy guarantees as adding Laplace noise to the counts. Insertion and deletion of z cases work similarly to adding Laplace noise, with rounding negative results after noise injection up to 0. However, L contains strictly more information than the information represented by CT, e.g., the timestamp and the case IDs. An attacker can single out the individual using their timestamps (attack h_2). At this step, we assume having a fixed timestamp while anonymizing the case variants distribution, and we discuss the timestamp anonymization later in Sect. 5.1.6. Also, the attacker can guess the new injected cases through their case IDs. For example, if the actual case IDs range from 0 to 10, and we start the injected new cases from case ID= 11, the attacker will clearly see which cases are duplicates. In this chapter, we shuffle the cases and generate new case IDs for all the cases in the event log. More formally, the final result is a multiset of cases, equivalent to the counts of a corresponding CT. ■

Timestamp Anonymization. In this section, we consider timestamp anonymization. We need to add enough noise to the timestamps to prevent attacker success in

h_2, and h_1, since timestamps may potentially leak which traces are real and which are not (new injected cases by sampling). We note that correlations between timestamps are difficult to handle in terms of guessing advantage. The main problem is that the effect of correlated times would greatly depend on how they are correlated, and in some cases, even a little leakage of t_2 may leak everything about t_1. For example, let $t_1 \in \{0,1\}$ and $t_2 \in \{0,...,1023\}$ be distributed uniformly. We have the prior knowledge $P_1 = 1/2$, and $P_2 = 1/1024$. Suppose that, after seeing the noisified output, the attacker constrained his view of t_2 to $\{0,...,511\}$. There are now 512 possible choices for t_2, so $\delta = 1/512 - 1/1024 = 1/1024$, which is very small. However, when t_1 is the highest bit of t_2, the value of t_1 would be leaked. This particular correlation of times is unlikely in practice, but it demonstrates the problem in general.

To this end, we convert a timestamp t_k to a time difference $dt_k := t_k - t_{k-1}$ of sequential events, which is the duration of time that an individual has spent in a transition from one event to the next one. The timestamp at time t_0 is the starting time st, which needs to be published as well to make it possible to reconstruct the actual timestamps. st can be converted to the time difference between the start time of a case and the start time of the log. We obtain the result for linearly correlated time differences (e.g., increasing the duration of the first event by a minute increases every other event at most by one minute). To this end, we scale ε_k by the length of the trace. Prop 4.3.4 proposes a mechanism that anonymizes the time differences.

Event Log Anonymization. We now provide a proof of correctness of Alg. 2.

Lemma 5.1.2. *Let L and δ be an event log and a maximum guessing advantage threshold. Log L' = Alg.2(L,δ) fulfills the two properties in Def 4.1.1 (i.e., L' is ε-differentially private), and L' fulfills requirement **UR1b** and the difference between the real and the anonymized time values is minimal (c.f. UR2).*

Proof. We first show that L' fulfills the two properties in Def 4.1.1.

(1) In Prop. 5.1.1, we proved that it suffices to take

$$\varepsilon_d = -\ln\left(\frac{P}{1-P}\cdot\left(\frac{1}{\delta+P}-1\right)\right)$$

to anonymize the case variants of an event log in order to provide differential privacy guarantees for the desired guessing advantage upper bound δ. Algorithm 2 annotates the event log using DAFSA transitions (line 3), and computes the contingency table CT of the log. After that, the algorithm uses the above ε_d to sample cases (replication and deletion of cases) (lines 18 and 21). Given that Prop. 4.3.2 and Prop. 5.1.1 prove that replicating and deleting cases with a random sample size z from Laplace distribution $Lap(1/\varepsilon)$ provides differential privacy guarantees. Algorithm 2 provides differential privacy guarantees for the case variants anonymization;

(2) In Prop. 4.3.4, we proved that it suffices to take

$$\varepsilon_k = -\frac{\ln\left(\frac{P_k}{1-P_k}\cdot\left(\frac{1}{\delta+P_k}-1\right)\right)}{r_k}$$

to provide differential privacy guarantees w.r.t timestamp for the desired guessing advantage upper bound δ. Given that Algorithm 2 adopts the above ε to anonymize timestamp each event (line 27). Hence, Algorithm 2 provides differential privacy guarantees for the event log anonymization w.r.t. timestamp.

We now show that L' fulfills the two requirement **UR1b** and the difference between the real and the anonymized time values is minimal.

1. Algorithm 2 does not create new case variants. It only replicates or deletes existing variants (cf. lines 18 and 21). Hence, it fulfills requirement **UR1b**;
2. In Prop. 4.2.2, we proved that the ε_k as defined in Eq (4.6) is the maximum within the upper bound δ, so Algorithm 2 assures that the difference between the real and the anonymized time values is minimal.

■

5.1.7. Timestamp Compression

This chapter assumes that the start timestamps of the first and the last case in the original log are public. However, the timestamp anonymization introduces time shifts, making some cases happen before the original first timestamp or after the last timestamp of the log. To reduce the impact of time shifts over the log, we perform timestamp compression as post-processing of the anonymized log.

Proposition 5.1.3 (Differential Privacy under Post-processing [6])**.** *A post-processing algorithm P of an event log L gives ε-differential private event log, if and only if it has been applied to the output of an algorithm A that gives an ε-differential private event log.*

Proof. The proof of Prop. 5.1.3 is in [6] (cf. Proposition 2.1) ■

Given the ε-DP log, the post-processing of $M(L)$ output is differentially private. In this chapter, we use the public information, the start timestamp of the first and the last cases in the event log, to post-process the anonymized event log. We perform timestamp compression to make the cases' timestamp fall between the original start timestamp of the first and the last cases in the event log. We multiply the relative time values with a compression factor. We define the compression factor as:

$$\textit{Compression Factor} = \frac{\textit{Original Range}}{\textit{Anonymized Range} + \textit{Orignal Range}} * \frac{1}{2}, \tag{5.1}$$

where the original range is the difference in days between the start timestamp of the first and the last cases in the original event log, and the anonymized range is the difference in days between the start timestamp of the first and the last cases in the anonymized event log. Table 10 shows the anonymized version of the input event log in Table 4 with a guessing advantage threshold δ=0.3.

Lastly, we generate new Case IDs for the anonymized log cases that do not link to the original log. Also, we reorder the events in the event log based on their new timestamps.

Table 10. Differentially Private Event Log with δ=0.3

Case ID	Activity	Timestamp
66d19fc868978d2fc1e	D	2020-08-08 04:26:24
66d19fc868978d2fc1e	A	2020-08-08 07:03:57
66d19fc868978d2fc1e	E	2020-08-08 07:43:20
66c81c1d1a9773464aa	D	2020-08-09 12:16:42
66d19fc868978d2fc1e	C	2020-08-09 13:33:07
66c81c1d1a9773464aa	A	2020-08-09 14:54:15
66c81c1d1a9773464aa	B	2020-08-09 15:53:27
c4247e4c1e9292166cd	A	2020-08-09 18:44:10
c4247e4c1e9292166cd	E	2020-08-09 19:23:33
efd62407e7f2b016e33	A	2020-08-09 19:34:35
efd62407e7f2b016e33	B	2020-08-09 20:00:23
efd62407e7f2b016e33	C	2020-08-10 00:49:12
66c81c1d1a9773464aa	C	2020-08-10 04:41:39
64b00e4dfc2b2145e17	D	2020-08-10 05:17:48
64b00e4dfc2b2145e17	A	2020-08-10 07:55:20
64b00e4dfc2b2145e17	B	2020-08-10 08:44:53
64b00e4dfc2b2145e17	C	2020-08-11 00:59:32
c4247e4c1e9292166cd	C	2020-08-11 06:16:25
cde714d92c42217500a	A	2020-08-11 07:02:37
cde714d92c42217500a	B	2020-08-11 07:33:55
cde714d92c42217500a	C	2020-08-11 15:26:28

5.2. Software Implementation

In this section, we present Amun, a tool for event log anonymization. Amun allows event log publishers to control the level of privacy in the anonymized event log. In other words, a user needs to upload an event log and specify the maximum level of acceptable guessing advantage. Amun applies differential privacy to prevent singling out an individual using a subtrace of their process execution. It does so by grouping individual traces based on their prefixes and suffixes. This grouping enables Amun to perform in a parallel manner. Following, we give an overview of

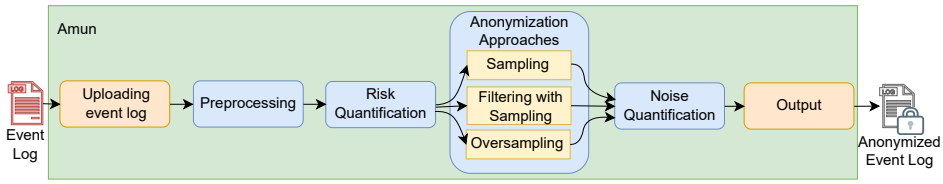


Figure 7. Overview of Amun

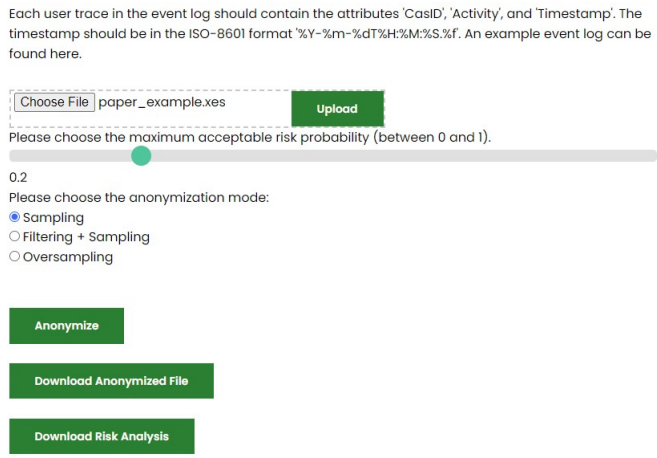


Figure 8. Upload an event log and anonymize it using a selected approach

the Architecture and implementation of Amun.

5.2.1. Functionality

Figure 7 gives an overview of Amun's components. Below, we summarize the functionality of each component of Amun. Amun's detailed explanation and evaluation are presented in Sect. 4.3 and Sect. 5.1.

Input. Figure 8 presents the upload page of the web application. The event log publisher uploads their event log to Amun as either an XES (eXtensible Event Stream) or CSV (Comma Separated Value) file. Amun requires the event log to have at least a column representing the case ID, a column representing the activity instance, and a column that records the timestamp executing each activity. Then, the user sets the maximum acceptable risk probability (δ) using the slider, selects the anonymization method (sampling, oversampling, or filtering), and clicks Anonymize.

Preprocessing and risk quantification. Once the user clicks Anonymize, Amun starts processing the file. The first step is to establish a representation that helps to quantify the re-identification risk attached to releasing each event in the log. To this end, Amun represents the input event log as a lossless representation, namely a Deterministic Acyclic Finite State Automata (DAFSA) [14]. Next, Amun annotates each event log with its DAFSA transition, as explained in Sect. 5.1.

Then, for each event, Amun estimates the prior knowledge P_k, which represents the re-identification risk before publishing the log, and the posterior knowledge P'_k, which means the re-identification risk after publishing the log. A detailed explanation of this risk quantification is presented in Sect. 5.1.

Anonymization Methods. Amun offers the user three different anonymization approaches. All the approaches guarantee that the customers in the anonymized log will not be singled out using a subset of their trace variants or the timestamp of executing their activities. All the approaches provide differential privacy guarantees [6] by injecting noise, quantified by the differential privacy parameter ε, from the control flow perspective, representing user traces in the log and the timestamp perspective. The first approach is oversampling Sect. 4.3 and Sect. 5.1. Oversampling requires that the set of trace variants in the input log and the anonymized log are the same. To this aim, Amun applies the approach presented in Sect. 4.3 and Sect. 5.1. The second approach is Sampling. The sampling approach anonymizes the event log so that the anonymization does not add new trace variants in the log, and the difference between the real and the anonymized timestamp is minimal. Amun applies the sampling approach presented in Sect. 5.1. Some event logs may contain very unique user traces, resulting in large noise injection to achieve differential privacy guarantees. Therefore, Amun applies the filtering approach presented in Sect. 5.1.

Noise Quantification and Injection. Given the estimated re-identification risk per event, Amun estimates the suitable ε value. We draw noise from Laplacian distribution and inject noise for both the control flow and time perspectives. This step is performed for each event independently.

Output. Once the event log anonymization is finished, the anonymized event log will be available for download. Amun downloads the anonymized log in the same format as the original log. Amun offers to download the risk quantification of each activity instance in the log as a CSV file. The risk quantification per each activity instance is a column called original risk, which represents the re-identification risk of releasing the event log before the anonymization. Amun anonymizes only the three columns: case ID, activity label, and timestamp. Amun drops the other attributes from the anonymized log.

5.2.2. Maturity and Availability

Amun is developed as a React web application and an API for ease of use. To enable quick trials by the users, Amun is available as a cloud service that can be found at http://amun.cloud.ut.ee. The current server deployment accepts event logs with sizes up to 5 MB. Amun is available as a docker image. The image and its installation steps can be found at https://github.com/Elkoumy/amun/tree/amun-flask-app. Also, Amun is available as a python package and can be integrated into other process mining tools. The source code and the installation steps can be found at https://github.com/Elkoumy/amun. A screencast that describes the tool is

available on YouTube at https://youtu.be/1dxaCNE9WHk.

5.3. Evaluation

To address the RQ1 under requirements **UR1b** and **UR2** , the proposed method injects differentially private noise in two ways: (i) by sampling and filtering some of the traces in the log; and (ii) by altering the event timestamps. The noise injection and case filtering affect the utility of the anonymized logs. We measure the effect of anonymization on the utility by comparing the anonymized logs against the original ones. We compare the performance of different design choices (the design choices presented in this chapter and in Chapter 4). Also, we compare the proposed approach against the state-of-the-art.

Accordingly, we define the following questions:

Q1. Does loosening the requirement from UR1a to UR1b preserve a higher utility of the anonymized event log?

Q2. Does the proposed approach outperform the state-of-the-art baselines in terms of the output utility?

Q3. What is the difference between different design choices and the state-of-the-art in terms of computational efficiency?

5.3.1. Evaluation Measures

To measure the quality of the anonymized case variants, we use Jaccard Distance (JD) [89]. It is efficient for sparse vectors and has been adopted in existing work [90]. JD is a metric that measures the similarity and diversity of two sets. It is calculated by dividing the size of the intersection by the size of the union of the two sets. $JD(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$. This chapter compares the set of case variants in the anonymized log against the original log.

Given a log, a typical output of process mining tools is the DFG. To measure the utility loss of the anonymization on the DFG, we compare the DFG resulting from the anonymized log against the original one. To measure the difference between two DFGs, we use the *Earth Movers' Distance* (EMD) [91]. The EMD between two distributions u and v is the minimum cost of transforming u into v. The cost is the distribution weight that needs to be moved, multiplied by the distance it needs to move. Formally:

$$EMD(u,v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x-y| d\pi(x,y), \tag{5.2}$$

where $\Gamma(u,v)$ is the set of distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are u and v.

Optimal anonymization is an NP-hard problem [92]. Hence, increasing the log size makes it unpractical to perform anonymization within sufficient execution time [48]. Consequently, we conduct a wall-to-wall run time experiment to assess

the efficiency of the method. We measure the time between reading the input XES file and the generation of its anonymized version.

5.3.2. Event Logs

To answer our research questions, we rely on the real-life event logs publicly available at 4TU Centre for Research Data[3] as of February 2021. We considered the logs mentioned in Table 11. The selected logs contain the process execution of different domains, e.g., government and healthcare. From the set of available logs, we excluded the event logs that are not business processes (e.g., "Apache Commons", "BPIC 16" logs, "Junit 4.12"). Also, we exclude the set of event logs "coSeLog" as they are a pre-processed version of BPI challenge 15. Finally, we select a single log for each set of logs in BPI challenges 13, 14, 15, 17, 20.

5.3.3. Experiment Setup

We implement the proposed model as part of a prototype, namely Amun[4]. We run the experiment on a single machine with AMD Opteron(TM) Processor 6276 and 32 GB memory. We time out any experiment at 24 hours. Also, in our experiment, we consider only the end timestamp to calculate the relative time of an event for simplicity, and the same approach is still valid to apply DP. Further, we keep only the three attributes in every event log: case ID, Activity, and timestamp.

We evaluate and compare the different design choices of our approach. We evaluate the oversampling presented in the conference version of this chapter (Sect. 4.3), the proposed approach using all the proposed steps in Sect. 5.1 (filtering risky cases then sampling), and the proposed approach without the third step (sampling without filtering). We use the JD and EMD to compare the anonymized event log against the original for all the design choices.

We compare the proposed approach against the state-of-the-art. The studies that consider PPPM from the case perspective are [12,60,64–66]. In our comparison, we do not include the work in [12, 60] because the parameters' interpretation of the k-anonymity privacy model is different from the DP model. The studies [64–66] adopt DP. Mannhardt et al. [64] anonymize two types of queries: the query "frequencies of directly-follows relations" and "frequencies of trace variants". The output of the anonymization of [64] is not an event log. PRIPEL [65] anonymizes the event log while adopting the trace variant queries anonymization that has been proposed in [64]. We compare the proposed approach against [65]. SaCoFa [66] anonymizes the case variant queries. The output is an event log without the time attribute. We include SaCoFa [66] only in the case variant experiments. PRIPEL and SaCoFa take three input parameters, namely ε, k, and N. To select the parameters' values, we run several experiments for different values of the pruning parameter k (0.5%,

[3] https://data.4tu.nl/

[4] https://github.com/Elkoumy/amun

Table 11. Descriptive Statistics of Event Logs

event log	# Traces	# Tasks	# Events	# DF Rel.	Case Variant	Trace Length		Case Duration		
						Min	Max	Min	Max	Avg
*BPI*12 [93]	13087	23	262200	116	4366	3	175	1.85 s	4.51 m	1.23 w
$BPI13_i$ [94]	7554	4	65533	16	1511	1	123	inst.	2.11 y	1.73 w
$BPI14_i$ [95]	46616	39	466737	497	22632	1	178	14 s	1.07 y	5.07 d
$BPI15_1$ [96]	1199	398	52217	495	1170	2	101	8.56 h	4.07 y	3.15 m
*BPI*17 [97]	31509	24	1202267	181	3942	10	180	3.35 m	9.4 m	3.13 w
*BPI*18 [98]	43809	14	2514266	499	28457	24	2973	3.74 m	2.77 y	11.03 m
*BPIC*19 [99]	251734	42	1595923	498	11973	1	990	2 ms	70.33 y	2.35 m
$BPI20_r$ [100]	7065	51	86581	500	1478	3	90	12.61 h	3.26 y	2.87 m
*CCC*19 [101]	20	29	1394	149	20	52	118	11 m	1.01 d	1.73 h
CredReq [102]	10035	8	150525	9	1	15	15	3.5 h	5 d	22 h
Hospital [103]	1143	624	150291	903	981	1	1814	inst.	3.17 y	1.06 y
Sepsis [104]	1050	16	15214	115	846	3	185	2.03 m	1 y	4 w
Traffic [105]	150370	11	561470	77	231	2	20	3 d	12 y	11 m
Unrine. [106]	1650	10	6973	25	50	2	35	10.1 m	2.32 y	3.7 w

1%, and 5% of the cases), and we select the best results. For the maximum trace length, we set N to the average trace length of the log.

PRIPEL accepts a single ε value for both the trace variant anonymization and the timestamp attribute anonymization, and SaCoFa adapts a single ε value for all the events. On the contrary, our proposed approach uses different ε values. Consequently, we use the average value of the estimated ε values in our approach as input to PRIPEL and SaCoFa. We do that in two different settings. We run an experiment using the ε values estimated for the trace variant anonymization (i.e., ε_d, as explained in Sect. 5.1.4), and we evaluate the output using the JD. Also, we run another experiment using the ε values estimated for the timestamp anonymization (i.e., ε_t, as explained in Sect. 5.1.4), and we evaluate the output using the EMD. We use SaCoFa with only the first setting because it anonymizes the case variant query only.

5.3.4. Results

Table 12 shows the experimental results measured by JD. The minimum JD for each event log, given an input δ, is highlighted in bold. A "-" indicates that the approach runs out of memory (32 GB) or times out (24 hours). ε_d refers to the ε estimated from the guessing advantage to anonymize the sequence of events. We exclude the oversampling setting (c.f. Chapter 4) from this comparison, as it does not change the input case variants. The proposed approach using sampling without filtering (labeled $Amun_s$) has the minimum JD. The second-smallest distance is the proposed approach with the setting of both filtering and sampling (labeled $Amun_f$) because filtering removes more cases, which is penalized by the JD. The JD of PRIPEL and SaCoFa are significant because they generate new case variants and delete case variants. For some logs, both PRIPEL and SaCoFa runs out of memory.

The utility loss differs across logs. We see a decrease in the utility loss for structured logs such as Credit Requirement, Unrineweginfectie, and Traffic Fines. For example, the output of $Amun_s$ for Unrineweginfectie has a maximum JD of 0.07, and for the same log, $Amun_f$ has a maximum distance of 0.09. This difference is because $Amun_f$ filters out cases to reduce the timestamp noise injection. PRIPEL has a JD of 0.9, and SaCoFa has a JD= 0.89. That happens due to infrequent case variants trimming.

With anonymizing unstructured event logs, which is more challenging due to the uniqueness of cases, we see an increase in utility loss. For instance, the output of $Amun_s$ for Sepsis cases log has a maximum JD of 0.144, and for the same log, $Amun_f$ has a maximum distance of 0.88. The increase of the utility loss is significant with $Amun_f$ because it filtered out 580 case variants, in contrast to $Amun_s$, which filtered out only 87 case variants. For the same log, PRIPEL has a maximum JD of 0.9969. That happens because PRIPEL filtered out 839 case variants. SaCoFa has a maximum JD = 0.9598 because it filtered out 787 case variants and added new 19 case variants (false positives).

Table 12. Jaccard Distance for the output of different anonymization approaches. A "-" means that the approach ran out of memory or timed out.

Log Name	δ	ε_d	$Amun_s$	$Amun_f$	PRIPEL	SaCoFa
BPIC12	0.2	0.8100	**0.1618**	0.8681	0.9995	0.9931
	0.3	1.2380	**0.1140**	0.9005	0.9995	0.9926
	0.4	1.7000	**0.0612**	0.9187	0.9995	0.9928
BPIC13	0.2	0.8100	**0.12**	**0.12**	0.9966	0.9760
	0.3	1.2380	**0.05**	0.17	0.9966	0.9757
	0.4	1.7000	**0.02**	0.53	0.9964	0.9770
BPIC14	0.2	0.8100	**0.1470**	0.1581	-	0.9997
	0.3	1.2380	**0.0950**	0.1051	0.9999	0.9997
	0.4	1.7000	**0.0500**	0.0949	0.9999	0.9998
BPIC15	0.2	0.8100	**0.2685**	0.3053	-	-
	0.3	1.2380	**0.1750**	0.4455	0.9999	-
	0.4	1.7000	**0.0996**	0.5396	0.9997	-
BPIC17	0.2	0.8100	**0.0618**	0.6956	0.9997	0.9915
	0.3	1.2380	**0.0276**	0.9214	-	0.9916
	0.4	1.7000	**0.0130**	0.9216	0.9997	0.9933
BPIC18	0.2	0.8100	**0.2024**	0.6546	0.9999	0.9984
	0.3	1.2380	**0.1283**	0.6913	-	0.9989
	0.4	1.7000	**0.0689**	0.8055	-	-
BPIC19	0.2	0.8100	**0.1903**	0.2385	-	0.9957
	0.3	1.2380	**0.1385**	0.1977	-	0.9960
	0.4	1.7000	**0.0852**	0.2239	-	0.9962
BPIC20	0.2	0.8100	**0.1129**	0.1319	0.9948	0.9572
	0.3	1.2380	**0.0343**	0.0864	0.9948	0.9592
	0.4	1.7000	**0.0222**	0.1432	0.9948	0.9835
CCC19	0.2	0.8100	**0.1564**	0.9498	-	-
	0.3	1.2380	**0.2798**	0.7403	-	-
	0.4	1.7000	**0.0**	0.8838	-	-
CredReq	0.2	0.8100	**0.0**	**0.0**	**0.0**	**0.0**
	0.3	1.2380	**0.0**	**0.0**	**0.0**	**0.0**
	0.4	1.7000	**0.0**	**0.0**	**0.0**	**0.0**
Hospital	0.2	0.8100	**0.2849**	0.3269	0.9999	-
	0.3	1.2380	**0.2177**	0.2203	0.9998	-
	0.4	1.7000	**0.1295**	0.1579	0.9998	-
Sepsis	0.2	0.8100	**0.1437**	0.4060	0.9962	0.9485
	0.3	1.2380	**0.1226**	0.7185	0.9964	0.9556
	0.4	1.7000	**0.0340**	0.8820	0.9969	0.9598
Traffic	0.2	0.8100	**0.0185**	0.0275	-	0.9757
	0.3	1.2380	**0.0042**	0.0270	-	0.9773
	0.4	1.7000	**0.0**	0.1830	-	0.9762
Unrine.	0.2	0.8100	**0.0728**	0.0852	0.9875	0.8923
	0.3	1.2380	**0.0**	0.1123	0.9938	0.9006
	0.4	1.7000	0.0728	**0.0**	0.9875	0.9029

Table 13. Earth Movers' Distance for the output of different anonymization approaches. A "-" means that the approach ran out of memory or timed out.

Log	δ	ε_t	EMD Freq				EMD Time			
			$Amun_s$	$Amun_f$	$Amun_o$	*PRIPEL*	$Amun_s$	$Amun_f$	$Amun_o$	*PRIPEL*
BPIC12	0.2	1.48	**331.02**	653.36	2301.62	946.9	40.30	**8.08**	192.13	25.75
	0.3	2.00	**212.64**	742.39	1597.79	966.99	20.32	**13.54**	107.96	26.7
	0.4	2.47	**142.37**	785.87	1275.43	966.88	**12.68**	16.74	72.56	26.7
BPIC13	0.2	1.49	1131.91	**1053.45**	7613.27	3771.09	778.18	811.51	3343.92	**197.41**
	0.3	1.99	840.55	**258.45**	5417.64	3792.55	592.36	307.26	2055.23	**197.38**
	0.4	2.44	**558.45**	2450.45	4296.73	3775.82	486.86	**75.13**	1751.96	197.19
BPIC14	0.2	1.25	429.60	**395.24**	1905.69	531.42	132.53	130.00	577.32	**10**
	0.3	1.74	298.86	**281.43**	1333.95	-	82.02	**76.37**	321.72	-
	0.4	2.27	208.55	**179.94**	1012.62	-	**47.12**	50.61	178.08	-
BPIC15	0.2	0.42	20.71	**18.74**	80.61	-	5.68	**4.18**	22.47	-
	0.3	0.69	14.82	**7.93**	52.42	10.71	3.15	2.28	11.17	**0.8**
	0.4	0.99	12.60	**2.79**	39.09	-	2.46	**1.05**	8.87	-
BPIC17	0.2	1.81	**141.37**	1925.92	1159.08	2454.47	117.56	**75.98**	268.08	109.10
	0.3	2.21	**78.60**	2667.91	938.79	-	**95.07**	131.76	206.76	-
	0.4	2.66	**49.93**	2674.76	938.79	-	**79.33**	133.77	206.76	-
BPIC18	0.2	0.91	3775.468	**659.01**	17668.46	-	2176.17	**179.39**	3206.75	-
	0.3	1.31	2922.80	**1752.72**	12063.81	-	1319.69	**325.14**	2201.39	-
	0.4	1.72	**2372.08**	2812.65	9055.04	-	885.81	**517.59**	1647.15	-
BPIC19	0.2	2.96	946.99	**811.80**	4509.08	-	**524.18**	608.02	1513.92	-
	0.3	3.51	743.96	**572.71**	3094.05	-	**491.84**	498.59	1054.23	-
	0.4	4.00	612.48	**399.36**	2376.21	-	500.57	**360.54**	751.67	-
BPIC20	0.2	2.73	18.97	**18.69**	138.05	98.10	65.23	69.04	180.50	**23.94**
	0.3	3.27	14.88	**10.42**	99.76	-	44.46	**40.01**	109.90	-
	0.4	3.75	10.55	**2.24**	76.61	-	41.70	**35.78**	82.67	-
CCC19	0.2	0.23	12.78	**3.70**	30.81	-	**0.00**	**0.00**	**0.00**	-
	0.3	0.54	4.55	**2.23**	25.77	-	**0.00**	**0.00**	**0.00**	-
	0.4	0.85	5.54	**3.21**	16.64	-	**0.00**	**0.00**	**0.00**	-
CredReq	0.2	1.39	**0.00**	2.00	4.00	**0.00**	233.08	238.79	186.94	**0.00**
	0.3	1.80	**0.00**	**0.00**	3.00	**0.00**	203.94	182.39	220.17	**0.00**
	0.4	2.03	**0.00**	2.00	2.00	**0.00**	201.81	231.37	180.77	**0.00**
Hospital	0.2	0.21	81.40	**75.74**	406.45	-	**10.30**	11.33	55.44	-
	0.3	0.41	**57.91**	61.96	264.09	-	7.86	**7.42**	31.71	-
	0.4	0.60	**58.68**	63.93	210.40	-	6.07	**6.01**	20.70	-
Sepsis	0.2	1.31	56.84	**32.20**	427.64	118.02	8.97	**4.37**	61.50	8.68
	0.3	1.83	**28.46**	67.97	286.23	118.63	6.35	**3.60**	32.93	8.64
	0.4	2.37	**43.38**	101.64	232.50	118.71	**4.26**	6.48	26.61	8.64
Traffic	0.2	4.50	1.64	**0.90**	33.50	-	8250.42	**7253.29**	8627.86	-
	0.3	5.26	**0.61**	8.01	25.51	-	7767.32	**6720.80**	7081.06	-
	0.4	5.28	**0.00**	2951.50	21.00	-	**7182.96**	12788.57	7419.02	-
Unrine.	0.2	2.87	6.53	**5.53**	66.00	290	44.96	**39.34**	184.68	94.74
	0.3	3.41	**0.00**	2.87	53.67	290.47	**28.12**	33.27	103.67	94.73
	0.4	3.84	**1.47**	2.53	47.00	290.47	**21.89**	26.35	90.93	94.73

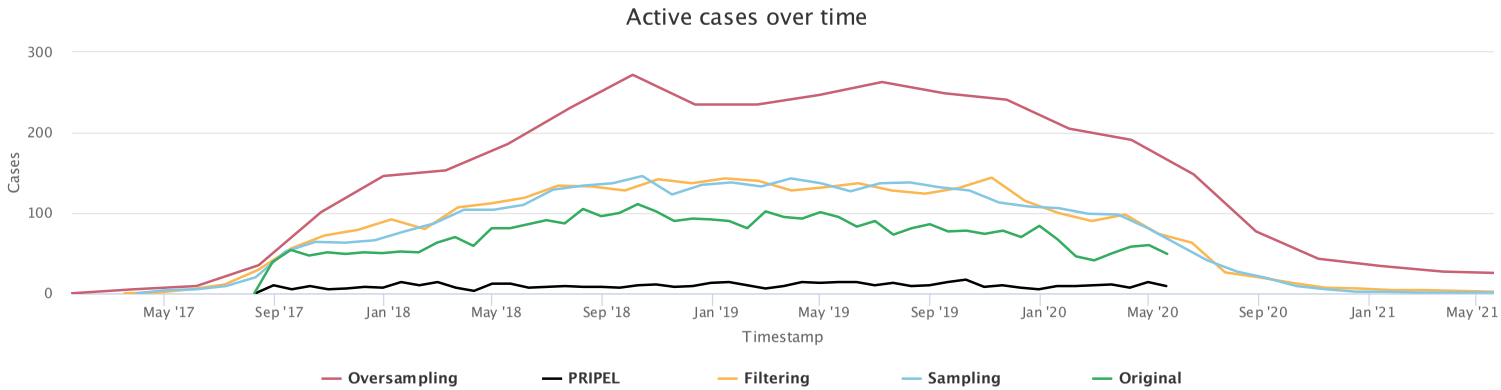
The second setting of our experiment is to measure the utility loss over the DFG. Table 15 shows the results for both the frequency and time annotated DFG. We use the total relative time between two activities to annotate the DFG. The time EMD distance is measured in terms of months. ε_t refers to the average ε_t value estimated by the proposed approach for timestamp anonymization. The best result for every input δ is in bold. $Amun_f$ outperforms other settings in the time EMD in most of the logs because case filtering decreases the needed noise to anonymize the timestamp, and hence $Amun_f$ has a lower utility loss.

Conversely, the $Amun_f$ has a lower frequency EMD than $Amun_s$, due to the decrease in the frequencies by case filtering. The oversampling setting (Chapter 4), labeled $Amun_o$, has the largest frequency and time EMD due to duplicating cases and dividing ε_t by the number of duplications. PRIPEL has a time EMD that is close to the filtering settings. However, the used ε_t with PRIPEL is the average estimated ε. Thus, the proposed approach can provide similar or better time EMD with stronger privacy metrics. The effect of using the same ε in PRIPEL for both the case variant and timestamp anonymization appears in the frequency EMD. The proposed approach has a lower frequency EMD than PRIPEL in all the logs.

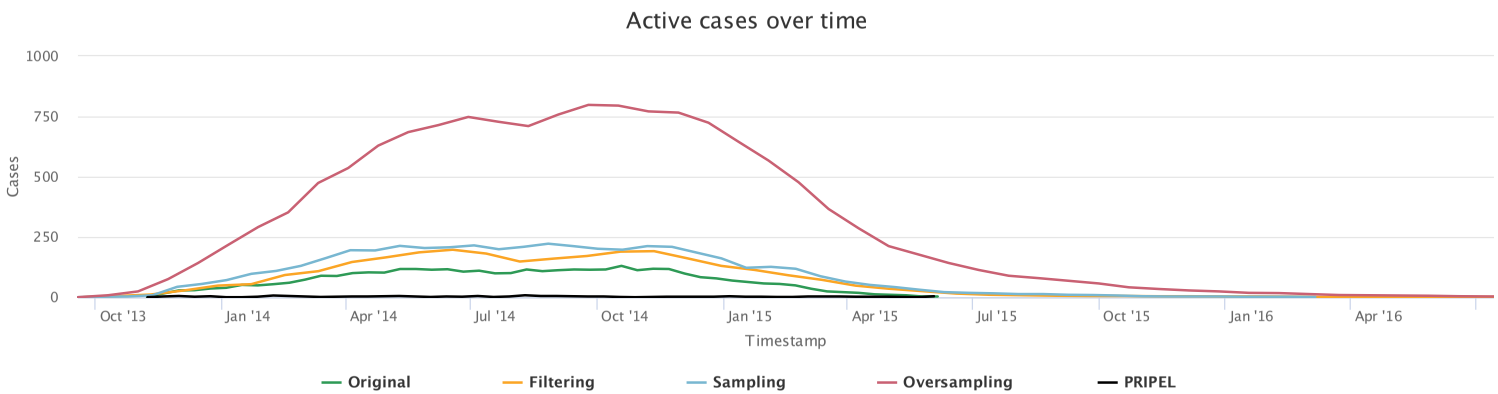
We compare the variant analysis of different design choices for structured and unstructured event logs for further analysis. Fig. 9 shows a variant analysis between Unrineweginfectie and Sepsis logs and their anonymized versions by different design choices and PRIPEL. All the design choices have their best utility for the Unrineweginfectie log, which has 1650 cases and only 50 case variants. Fig. 9(a) shows that $Amun_s$, $Amun_f$, and PRIPEL result in logs with close active cases over time to the original log. The false negatives in the anonymized logs are 1, 2 and 34 for $Amun_s$, $Amun_f$ and PRIPEL, respectively. However, $Amun_o$ adds more noise than other approaches, though it keeps the false negatives to zero due to oversampling.

Fig. 9(b) shows the active cases over time for the sepsis cases event log, with 1050 cases and 846 case variants. For both $Amun_s$ and $Amun_f$ the anonymized version of Sepsis cases has a closer behavior to the original log. $Amun_o$ generates more noise with unstructured event logs than structured logs. The anonymized log by PRIPEL has fewer case variants than the original log. The false negatives in the anonymized logs are 37, 317, and 839 for $Amun_s$, $Amun_f$, and PRIPEL, respectively.

We conduct a wall-to-wall run time experiment to assess the efficiency of the method. We measure the time between reading the input XES file and the generation of its anonymized version. The results are reported in Table 16, and the values are in minutes. The run time increases with case variants (as it contains more DAFSA transition groups). The execution times for logs with numerous events and with low δ values are in the order of hours, e.g., 2.13 hours for BPIC18 (2.5 million events) with δ=0.2 because the noise injection algorithm iterates multiple times over each transition (lines 15-23 in Algorithm 2), and the number of DAFSA states for this log is high (638,242 states). This shortcoming can



(a) Unrineweginfectie Active cases over time



(b) Sepsis Active Cases over Time

Figure 9. Variant Analysis comparison between Unrine. and Sepsis event logs and their anonymized versions, with $\delta = 0.2$, average $\varepsilon = 1.31$ for Sepsis, and average $\varepsilon = 2.87$ for Unrine. The figures are zoomed by 70%.

Table 14. Execution time experiment. The time is measured in minutes for an input $\delta = 0.2$. A "-" means that the approach ran out of memory or timed out.

event log	$Amun_f$	$Amun_s$	$Amun_o$	PRIPEL
BPIC12	4.44	7.38	12.50	24.35
BPIC13	1.11	1.09	3.90	4.20
BPIC14	44.57	43.83	101.40	-
BPIC15	4.38	4.45	8.40	-
BPIC17	6.88	10.80	17.40	1.46
BPIC18	128.32	321.59	542.50	226
BPIC19	96.89	52.73	87.50	-
BPIC20	2.64	2.52	4.10	3.18
CCC19	0.04	0.09	0.10	-
CreditReq	1.21	1.35	1.20	14.50
Hospital	11.27	10.65	34.10	-
Sepsis	0.87	1.00	1.90	0.05
Traffic	9.71	11.35	8.20	-
Unrine.	0.14	0.16	0.20	0.02

be tackled via parallelization, as the privacy quantification over each DAFSA transition is independent of others. Specifically, the step at line 4 of Algorithm 2 can be parallelized since the prior knowledge estimation is calculated per the DAFSA transition. Also, line 6, the ε_t estimation is estimated for each event separately; therefore, it can be computed in parallel. PRIPEL does not scale and runs out of memory for event logs that have numerous cases and events. The above experiments were all done using one single thread to avoid bringing additional variables (number of computing nodes and cores) into the experiments.

We acknowledge that the above observations are based on a limited population of logs (14). However, these logs were selected from a broader population of close to 50 real-life logs.

5.3.5. Evaluation Conclusion

The above experiments show that loosening the requirement from UR1a to UR1b preserve a higher utility of the anonymized event log. Specifically, the results in Table 12 put into evidence the consistently low utility loss of using sampling of events across the lossless representation of log to apply DP. Furthermore, the results in Table 15 and Fig. 9 show that using different ε values for case variant and timestamp anonymization leads to lower utility loss. Moreover, the use of personalized DP to estimate different ε per event leads to a stronger privacy guarantee with lower utility loss. The experiments also show that the proposed approach outperforms the state-of-the-art baselines in terms of output utility and computational efficiency.

5.4. Summary

This chapter proposed a concept of the differentially private event log and a mechanism to compute such logs. A differentially private event log limits the increase in the probability that an attacker may learn a suffix of an individual's trace given a prefix (or vice-versa) or the timestamp of activity in an individual's trace. To this end, we inject differentially private noise by sampling the traces in the log. This approach does not add case variants (cf. Sect. 5.1) and hence fulfills **UR1b**. To inject minimal noise to the timestamps of the event log, we quantify ε based on a technique that finds the maximum ε (minimum noise) that keeps the guessing advantage below δ.

The empirical findings show that the proposed approach is a step toward anonymizing event logs while preserving the utility of the process mining analysis. The proposed approach outperforms the baselines in terms of Jaccard distance and earth movers' distance, and generalization to all fourteen real-life event logs selected in the evaluation. Furthermore, the approach can process large size event logs in practical memory size (32 GB).

A limitation of the proposed method is that it anonymizes the timestamp without performing calendar anonymization, violating organizational rules (e.g., single activity at each timestamp). A possible avenue for future work is to model and include organizational rules while anonymizing logs.

6. DIFFERENTIALLY PRIVATE RELEASE OF EVENT LOGS: A SUBSAMPLING APPROACH

In this chapter, we address the research question **RQ1** under the utility requirement *UR1b: The anonymized event log must not introduce new case variants to the original log.*. As we presented above in Chapter 2, differentially private mechanisms work by injecting noise to achieve DP-guarantees. The holy grail of anonymization techniques in general, and DP-anonymization techniques in particular, is to achieve a low level of re-identification risk with low utility loss. Recent work in the field of DP has shown that the privacy guarantees of a differentially private mechanism can be amplified by applying it to a small random subsample of records [107]. This property is known as *privacy amplification*. The underpinning idea is that, since we inject less noise to the subsampled records, there is less utility loss overall. In this chapter, we hypothesize that a DP approach based on subsampling can achieve lower utility loss, for a given level of privacy guarantee, relative to existing DP-anonymization techniques for event logs, which are based purely on noise injection.

The contribution of this chapter is a DP-anonymization approach for event logs, namely Libra. Libra starts by filtering out case variants that, if disclosed, would lead to privacy breaches. It then extracts multiple Poisson subsamples and applies a DP mechanism to anonymize each subsample. The resulting differentially private subsamples are then combined to construct an anonymized log. Using the differential privacy composition theorem [108] and the privacy amplification results associated to *Renyi Differential Privacy* (RDP) [109], we then estimate the amplified ε' privacy guarantee provided by the resulting anonymized log.

This chapter is structured as follows. Sect. 6.1 presents the anonymization approach. Sect. 6.2 presents an empirical evaluation. Finally, Sect. 6.3 concludes and discusses future work. This chapter is derived from [16] and contains sentences or fragments of sentences from this prior publication.

6.1. Approach

We seek to anonymize an event log in such a way an attacker cannot single out an individual (e.g. a patient) in the anonymized event log. Accordingly, Libra applies differential privacy to the event log. To provide a better usage of a given privacy budget ε, Libra relies on privacy amplification by subsampling [107, 109].

More concretely, given an event log and a privacy budget ε, Libra produces a differentially-private event log in 5 steps as outlined in Fig.14. Some case variants may be very rare, and keeping them can lead to singling out individuals. Accordingly, the first step clips case variants below a clipping threshold C, estimated by the given privacy budget ε. The second step constructs a Poisson subsample of the event log. The third step is to anonymize subsamples. Some of the anonymized

cases may affect the utility. Therefore, Libra performs statistical post-processing of the anonymized subsamples to select the relevant traces. Libra repeats the above process to generate multiple subsamples while tracking the privacy budget by noise screening, and it composes the privacy budgets of the subsamples using Renyi Differential Privacy (RDP). Lastly, Libra combines the generated subsamples to construct the anonymized event log. This chapter adopts the same attack model presented in Sect. 4.1. In the following, we explain each step of Libra in turn. Furthermore, an algorithm is presented to formalize Libra's steps.

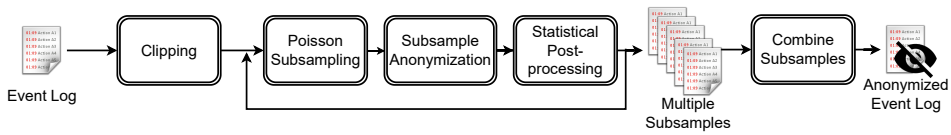


Figure 10. Approach

6.1.1. Clipping rare Cases

It is possible to single out individuals using their case variants, even if the variant is not unique, but exists a few times in the log. We call such a case variant a *rare case variant* [110]. Observing a rare case variant is problematic because it is executed for a group of few individuals, and observing such a trace may increase the attacker's confidence about this group of individuals. Chaudhuri et al. [110] define rare values (rare case variants) in a data sample w.r.t (ε, δ)-DP guarantee.

Lemma 6.1.1 (Rare case variants [110])**.** *A sample that provides (ε, δ)-differential privacy guarantee should not contain rare case variants that happen less than $1/\varepsilon \; log(\frac{2k}{\delta})$, where k is the number of case variants in the log.*

Since Libra provides privacy amplification via subsampling, any log sample that provides (ε, δ)-DP guarantee should not contain rare case variants [110]. If a rare case variant exists in the released log, it may lead to a privacy breach of $O(\frac{i\varepsilon}{log(2k/\delta)})$, where i is the number of rare cases in the log [110]. Libra clips case variants that happen less than $C = 1/\varepsilon \; log(\frac{2k}{\delta})$. It does so by filtering out all the trace instances of the case variants that happen $< C$.

Libra does not apply in the case that the log is a collection of unique traces (i.e., each case is of a unique case variant). The reason is that if every trace in the log is unique, then a sample of any size, even one, violates the privacy guarantees.

6.1.2. Event Log Subsampling

Our goal is to prevent an attacker from determining the existence of an individual in the anonymized log. To this end, we bring privacy amplification by subsampling [107, 109] into action.

Definition 6.1.1 (Subsampling)**.** Given an event log L of z traces, a subsample procedure selects a random set of traces from the uniform distribution of traces of L of size m. The ratio $\gamma := \frac{m}{z}$ is the sampling ratio of the subsample procedure.

Given Def. 2.4.1, DP works by adding or removing an item. Among other things, Poisson sampling works naturally with that property [109].

Definition 6.1.2 (Poisson Sample [109])**.** Given an event log L, a PoissonSample procedure returns a subset of traces of the event log $\{t_i|\sigma_i = 1, i \in [1..z]\}$ by sampling $\sigma_i \sim Ber(\gamma)$ independently for $i = 1,2,...,z$.

The above procedure is equivalent to performing subsampling without replacement with sampling rate$\sim Binomial(\gamma, z)$. A binomial distribution converges to a Poisson distribution with parameter λ at the limit of $k \to \infty, \gamma \to 0$ while $\gamma z \to \lambda$.

Libra applies Poisson Subsampling to the input event log and generates multiple subsamples. Each subsample contains entire traces rather than subtraces. That prevents the subsampling step from introducing case variants that do not exist in log. The next step explains the anonymization of each subsample.

Lemma 6.1.2 (Privacy Amplification [107, 109])**.** *If a mechanism M is (ε, δ)-DP, then a subsampled mechanism M $\circ$ PoissonSubsample provides (ε', δ')-DP with $\varepsilon' = log(1 + \gamma(e^{\varepsilon} - 1))$ and $\delta' = \gamma\delta$.*

6.1.3. Subsamples Anonymization

At this step, Libra provides ε-DP guarantee for each subsample. Privacy amplification is achieved by applying a DP mechanism to a subsample of the event log [107, 111].

Lemma 6.1.3 (Privacy Amplification [107, 109])**.** *If a mechanism M is (ε, δ)-DP, then a subsampled mechanism M $\circ$ PoissonSubsample provides (ε', δ')-DP with $\varepsilon' = log(1 + \gamma(e^{\varepsilon} - 1))$ and $\delta' = \gamma\delta$.*

Following, we introduce the concept of differentially-private subsamples (M $\circ$ PoissonSubsample) that prevents attacks h_1 and h_2.

Definition 6.1.3 (Differentially-Private Subsample)**.** Given a PoissonSample S as defined in Def. 6.1.2, the output of a mechanism $S' = M(S)$ is said to be ε-differentially private if: (1)it provides ε-differential privacy over the set of case variants; (2)the timestamp is ε-differentially private.

In order to anonymize the subsample, we could use any of the DP mechanisms in the literature [15, 64–66]. We use the mechanism presented in Chapter 4. It employs guessing advantage to estimate the differential privacy parameter ε. Then, it injects the noise over an event log representation called DAFSA. The noise injection for case variants is done by oversampling traces. In this chapter, we adopt the approach in Chapter 4 to sample traces (adding or deleting traces) instead of oversampling. Furthermore, we provide it with an ε value rather than a guessing advantage threshold. In the evaluation section, we measure the impact of the privacy

amplification on the utility loss, and we compare both the proposed approach and the one in Chapter 4.

To protect against linking a case to its individual via the cycle time of an activity (h_2), we inject noise drawn from a Laplace distribution quantified by a given privacy budget ε. Therefore, we anonymize two components of the timestamps: (1) case start time, which is the timestamp of the first event of the case; (2) the execution timestamp of every other event after that. In order to inject the noise to the start time, we introduce the relative start time of the case, which is the difference between the case start time and the first start time in the log. We inject random noise quantified by ε to the relative start time. Both the relative start time and the generated noise are measured in days.

On the other hand, we aim to protect the cycle time of each event, which is the difference between its execution timestamp and the execution timestamp of its successor event. We inject randomly generated noise quantified by ε to the cycle time. Both the cycle time and the generated noise are measured in minutes. After injecting noise as mentioned above, we transform the anonymized relative start time and cycle time to timestamps again via addition. At the end of this step, we have a differentially private subsample of the log (M $\circ$ PoissonSample).

6.1.4. Statistical Post-processing of SubSamples

Anonymization perturbs the utility of event logs. Libra selects statistically significant traces out of the anonymized log to provide higher utility of the differentially private event log. This selection process is a post-processing step to the anonymized subsamples. The result of a post-processing step of a differentially-private subsamples of a log provides the same differential privacy guarantees [6].

Given the differentially private Poisson subsample of the log, the post-processing of the selected subsample provides (ε, δ)-DP guarantees (c.f. Prop. 5.1.3. We use a statistical post-processing of the subsamples to pick the most relevant traces and reduce the utility loss. These relevant traces are assessed by a trace abstraction function $\psi : 2^{\omega} \rightarrow \chi$, where χ is the domain of information extracted from a trace [112]. This information can be related to the sequence of activities in a trace and the frequencies of activities. Bauer et al. [112] provide a log sampling mechanism that adopts a series of binomial experiments and picks traces that provide new information while being discovery sufficient with probability ρ. We adopt the work in [112] to pick the relevant traces out of the differentially private subsamples. A trace τ provides new information if its abstraction is far from the union of the abstraction, jointly derived from the subsamples. That should happen within a distance ω of the union abstraction. Thus, we consider the predicate [112]:

$$\pi^{\omega}(S'_e, \tau) \leftrightarrow d\big(\psi(\tau), \bigcup_{\tau' \in S'_e} \psi(\tau')\big) > \omega \ , \tag{6.1}$$

where π is the picked sample, S'_e is the differentially private subsample, and ψ

is the used abstraction. A subsample is discovery sufficient w.r.t. an abstraction ψ, a distance parameter ω, and probability ρ as follows:

Definition 6.1.4 (Discovery Sufficiency [112])**.** A DP-subsample $S'_e \subset S'$ is (ρ, ω, ψ)-discovery sufficient, if there is a newly picked trace $\tau : \tau \in (S' \setminus S'_e)$ that: $p_\pi(S'_e, \tau) = P(\pi(S'_e, \tau) = 1 < \omega$, where p is a probability measure.

At the end of this step, we have an (ε, δ)-differentially private subsample of the log that has statistically relevant traces.

6.1.5. Combining Subsamples

In order to construct an (ε, δ)-differentially private event log with number of traces as close as possible to the number of traces in the original log, z, we repeat the differentially private subsampling operation for a number of times equals $\eta = \gamma z$. The repetitive access of the log via DP-subsampling requires the composition of the multiple differentially-private mechanisms (one mechanism for each DP subsample).

Theorem 6.1.4 (Differential Privacy Composition [6])**.** *Let M_1 and M_2 be $\varepsilon_1, \varepsilon_2$-differentially-private mechanisms. Then, the combination of the mechanisms $M_{1,2}(L) = (M_1(L), M_2(L))$ is $\varepsilon_1 + \varepsilon_2$-differentially private.*

Proof. The proof of Theorem 6.1.4 is in [6] (c.f. Theorem 3.14). ■

Indeed, the privacy parameters ε, δ degrade due to the composition. One way to compose DP mechanisms easily and with tighter guarantees is using Renyi Differential Privacy (RDP) [113].

Definition 6.1.5 (Renyi Differential Privacy [113])**.** A mechanism M is said to be (α, ε)-RDP with order $\alpha \in (1, \infty)$ if for every neighboring event logs L, L′:

$$D_\alpha(M(L)||M(L')) = \frac{1}{\alpha - 1} log\left(E_{\theta\ M(L')}[(\frac{P_{M(L)}(\theta)}{P_{M(L')}(\theta)})^\alpha]\right) \leq \varepsilon.$$

Lemma 6.1.5 (From RDP to DP [113])**.** *A mechanism M that provides (α, δ)-RDP, also provides $(\varepsilon + \frac{log 1/\delta}{\alpha - 1}, \delta)$-DP for $\delta \in (0, 1)$.*

RDP converges to (ε,0)-DP at $\alpha \to \infty$ [111]. Mironov [113] provides an estimation of the equivalent ε of the RDP as a function of α. The estimated ε for the Laplace distribution equals:

$$\varepsilon_{Laplace}(\alpha) = \frac{1}{\alpha - 1}\ log\left(\frac{\alpha}{2\alpha - 1} e^{\frac{\alpha - 1}{b}} + \frac{\alpha - 1}{2\alpha - 1} e^{\frac{-\alpha}{b}}\right) \tag{6.2}$$

The privacy amplification via subsampling for RDP has been studied in [109, 111]. Zhu et al. [109] estimate the resulted ε' after Poisson subsampling and composition using RDP as:

$$\varepsilon'_{M\circ PoissonSubsample}(\alpha) \leq \frac{1}{\alpha-1} log\Bigg((1-\gamma)^{\alpha-1}(\alpha\gamma-\alpha+1) \binom{\alpha}{2}\gamma^2(1-\gamma)^{\alpha-2}e^{\varepsilon(2)} +3\sum_{j=3}^{\alpha}\binom{\alpha}{j}(1-\gamma)^{\alpha-j}\gamma^j e^{(j-1)\varepsilon(j)}\Bigg) \tag{6.3}$$

6.1.6. Event Log Anonymization Algorithm

We exploit the above observation as formalized in Alg. 3. The event log anonymization mechanism takes as input an event log L, an order of RDP α, a differential privacy parameter δ, and a sampling ratio γ. The algorithm estimates the ε used to draw noise from the Laplace distribution using Eq. 6.2. Then, it sets k to equal the count of case variants in L. The algorithm estimates the clipping threshold C as explained in Lemma 6.1.1. The algorithm uses the estimated C to filter out case variants that occur less than C from the log L. We set z to equal the number of cases in the filtered log $\hat{L}$. Then, the algorithm estimates η, the number of subsamples needed to construct an event log with as many cases as the filtered log $\hat{L}$. After that, the algorithm performs Poisson sampling over $\hat{L}$. Given the Poisson subsample S, the algorithm generates an anonymized sample S' as explained in Sec. 6.1.3. Then, it picks the statistically relevant traces from the anonymized sample S' as explained in Sec. 6.1.4. The algorithm generates η subsamples and combines them to generate L'. Also, the algorithm reports the amplified ε' after the composition of privacy budgets of subsamples using Eq. 6.3.

Alg. 3 employs a differentially private mechanism to anonymize every Poisson subsample and performs post-processing of subsamples to pick the relevant traces. Since the post-processing depends only on the anonymized samples, the output of the post-processing is differentially private, c.f., Prop. 5.1.3. Then, the algorithm uses differentially private composition of the anonymized subsamples to construct the anonymized event log. Thus, the algorithm's output is differentially private.

6.2. Evaluation

To address **RQ1**, the proposed approach injects noise to provide (ε, δ)-DP guarantees in two ways: (1) by filtering out rare case variants and sampling traces in the log; (2) by introducing time shifts to the event timestamps. Filtering and noise injection affect the utility of the anonymized logs. To measure this utility loss, we compare the anonymized event logs against the original logs using a distance measure. Also, we compare the performance (execution time) of the proposed approach against state-of-the-art baselines.

The evaluation reported below is driven by the following questions:

Algorithm 3: Event Log Anonymization Algorithm

Input: L: Event Log,
α: order of Renyi Differential Privacy,
δ: Differential Privacy parameter,
γ: Sampling Ratio.
Output: L': ε'-Differentially Private Event Log,
ε': the amplified differential privacy guarantee.
$\varepsilon \leftarrow$ EstimateEpsilon(α); ▷ Estimate $\varepsilon_{Laplace}$ using Eq. 6.2.
$k \leftarrow$ CountVariants(L); ▷ Set k to the count of case variants in L.
$C \leftarrow (1/\varepsilon)\ log(2k/\delta)$; ▷ Estimate clipping threshold based on Lemma 6.1.1.
$\hat{L} \leftarrow$ Filter(L, C); ▷ Filter out case variants with frequency below C.
$z \leftarrow$ CountCases($\hat{L}$); ▷ Set z to the count of cases in $\hat{L}$.
$\eta \leftarrow \gamma z$; ▷ Calculate the count of subsamples.
$L' \leftarrow 0$;
while $i < \eta$ **do**
    S= PoissonSample($\hat{L}, \gamma$); ▷ Perform Poisson subsample as defined in Def. 6.1.2.
    $S' \leftarrow$ Anonymize(S, ε); ▷ Anonymize the subsample as explained in Sec. 6.1.3.
    $S'_e \leftarrow$; StatisticalPost-processing(S'); ▷ Perform statistical post-processing to select relevant cases as explained in Sec. 6.1.4.
    $L' \leftarrow L' \cup S'_e$;
    i++ ;
end
$\varepsilon' \leftarrow$ EstimateComposition($\alpha, \varepsilon, \gamma$); ▷ Estimate ε' using Eq. 6.3.
return L', ε'

Q1. Does the proposed approach outperform the state-of-the-art baselines in terms of the output utility?

Q2. What is the difference between the proposed approach and the state-of-the-art in terms of computational efficiency?

6.2.1. Evaluation Measures

In our experiment, we use the evaluation measures presented in Sect. 5.3.1, i.e., the earth movers' distance and the wall-to-wall execution time. We exclude the usage of Jaccard distance because in the below experiments, we do not measure the fulfillment of the utility requirement UR1a.

6.2.2. Event Logs

In our evaluation, we rely on the same event logs presented in Sect. 5.3.2. However, we exclude the event logs that always return empty anonymized results, e.g., CCC19, as explained in Sect. 6.1.1. Also, we exclude the event logs that timed out for all the approaches under comparison, e.g., BPIC18 and BPIC19.

6.2.3. Experiment Setup

We implement the proposed model as part of Libra[1] prototype. We run the experiment on a single machine with AMD Opteron(TM) Proc 6276 and 32 GB RAM. We time out any experiment at 24 hours. Also, in our experiment, we consider only the end timestamp to calculate the relative time of an event for simplicity. The approach still holds DP guarantees for logs with start and end timestamps. We fix the parameters b=2, γ=0.05, and $\delta = 10^{-4}$, and we evaluate the approach for different values of α. For an empirical evaluation of the relation between γ, α, and ε under Poisson subsampling, we refer to [109].

We compare our approach against the state-of-the-art. The studies that address the same problem are [12, 15, 60, 65, 66]. We exclude the work in [12, 60] from our experiment because the interpretation of the k-anonymity parameters and DP are different. The studies in [15, 64–66] provide DP guarantees. The work in [64] anonymizes two types of queries, but the output is not an event log. PRIPEL [65] adopts [64] for case variant anonymization. We compare the proposed approach against [65]. SaCoFa [66] provides case variant anonymization, but does not consider timestamp anonymization. Therefore, we consider [66] only in EMD of frequency annotated DFG experiments. Both PRIPEL and SaCoFa take three input parameters namely, ε, k, and N. For the pruning parameter k, we run several experiments with different values of k(0.5%, 1%,5% of the cases), and we select the best result. For the maximum trace length N, we set N to the average trace length of the log. Amun (c.f. Chapter 4 and Chapter 5) anonymizes both the case variants and the timestamps. It accepts the guessing advantage probability δ, rather

[1] https://github.com/Elkoumy/Libra

than ε. Also, Amun uses personalized differential privacy which means different ε values for different activity instances. Therefore, we exclude Amun from the below comparison. We use the EMD to compare the anonymized log against the original one for the selected approaches. We compare the proposed approach against both PRIPEL and SaCoFa.

6.2.4. Results

Table 15. Earth Movers' Distance for the output of different anonymization approaches. A "-" means that the approach ran out of memory or timed out, and A "N/A" means the engine returns an empty log.

Log	α	ε'	EMD Freq			EMD Time	
			Libra	*PRIPEL*	*SaCoFa*	*Libra*	*PRIPEL*
BPIC12	2	0.04	1036	-	**1007**	**19315**	-
	10	0.37	**997**	-	1007	**19228**	-
	100	4.7	**1005**	1230	1007	**19224**	**19224**
BPIC13	2	0.04	**3847**	4954	4947	**145272**	197564
	10	0.37	**3586**	4952	4948	**132827**	197564
	100	4.7	**3492**	4952	4952	**127752**	197564
BPIC14	2	0.04	**494**	-	544	**7088**	-
	10	0.37	**466**	-	544	**6970**	-
	100	4.7	**464**	-	507	**6944**	-
BPIC17	2	0.04	**1563**	-	2785	**48254**	-
	10	0.37	**1349**	-	2789	**41190**	-
	100	4.7	**1333**	-	2790	**40864**	-
BPIC20	2	0.04	**91**	128	125	**15602**	24422
	10	0.37	**82**	128	127	**13920**	24422
	100	4.7	**79**	128	127	**13504**	24430
Hospital	2	0.04	N/A	-	**35**	N/A	-
	10	0.37	**35**	-	**35**	**2507**	-
	100	4.7	**35**	**35**	**35**	2507	**2469**
Sepsis	2	0.04	N/A	**75**	117	N/A	**4644**
	10	0.37	123	**120**	121	**6218**	**6218**
	100	4.7	123	**120**	**120**	**6218**	**6218**
Unrine.	2	0.04	**120**	298	**42**	**58576**	68245
	10	0.37	**95**	292	**56**	**49005**	68234
	100	4.7	**81**	292	**64**	**37381**	68234

We measure the EMD between the DFG of the anonymized log and the DFG of the original log. We report the differences in terms of the frequency and time (measured in hours). Table 15 shows the experimental results using the EMD distance. α refers to the RDP parameter, and ε' refers to the equivalent DP parameter after the amplification and composition. The best result for every input α is in bold. Libra outperforms the state-of-the-art baselines in most of the logs, because the privacy amplification reduces the amount of injected noise, and hence achieves a lower utility loss.

Conversely, Libra outperforms both PRIPEL and SaCoFa over the frequency EMD because for a given ε'-DP guarantee, Libra needs an $\varepsilon > \varepsilon'$ due to privacy amplification. On the other hand, in event logs with many rare cases (Lemma 6.1.1) such as Sepsis and Urineweginfectie logs. Libra does not outperform the state-of-the-art because it adopts clipping to get rid of rare case variants. Due to clipping,

Table 16. Execution time experiment. The time is measured in seconds for an $\varepsilon' = 0.37$. A "-" means that the approach ran out of memory or timed out.

event log	*Libra*	*PRIPEL*
BPIC12	90	-
BPIC13	49	306
BPIC14	323	-
BPIC17	212	-
BPIC20	70	330
Hospital	90	-
Sepsis	10	24
Unrine.	9	2

Libra sometimes return an empty log, e.g., for Sepsis log with $\alpha = 2$ and Hospital log with $\alpha = 2$. Also, Libra has a lower utility loss over the anonymized timestamps (measured by EMD time), which happens due to privacy amplification.

We evaluate the processing efficiency of the proposed approach via a wall-to-wall run time experiment. Table 16 presents the experiments for $\alpha = 10$ ($\varepsilon' = 0.37$). The time is measured in seconds. We exclude SaCoFa from this experiment as it provides only case variant anonymization. The run time increases with the increase of the log size. The above experiments have been performed using a single thread to avoid adding other variables to the experiments.

We acknowledge that the proposed approach is not suitable for event logs with many rare case variants (unstructured event logs). In such a case, the approach filters out most of the cases, and sometimes it returns an empty event log. Also, the above observations are based on a limited population (8 event logs). However, we selected the logs from a broader real-life event log population.

6.3. Summary

This chapter proposed a differentially private mechanism to anonymize event logs for process mining. While previous proposals in this field rely purely on noise injection, the approach proposed in this chapter additionally employs subsampling to achieve stronger privacy guarantees with the same level of utility loss, or conversely, less utility loss for the same privacy guarantee (privacy amplification). The empirical evaluation shows that the privacy amplification effect leads to significant reductions of utility loss, particularly when it comes to anonymizing the frequency of distribution of case variants in a log (i.e. control-flow anonymization) and to a lesser extent when it comes to anonymization of event timestamps.

7. SECURE MULTI-PARTY COMPUTATION FOR INTER-ORGANIZATIONAL PROCESS MINING

In this chapter, we address *RQ2: How to enable process mining for interorganizational business processes without requiring the involved parties to share their private event logs or trust a third party?*

In response, we propose an architecture for process mining based on *secure multi-party computation* (MPC) [41]. In essence, MPC aims at the realization of some computation over data from multiple parties, while exposing only the result of the computation, but keeping the input data private. We consider the setting of an MPC platform, where the involved parties upload their event logs to a network of compute nodes. Before the upload, secret sharing algorithms locally split each single data value into different parts (i.e., shares) that are then stored at different nodes. Since each share does not provide any information about the original data, the uploaded event log is encrypted and exposed neither to the platform operator nor other involved parties. Nonetheless, the MPC platform enables the computation over the encrypted data through protocols for result sharing among the nodes.

We realize the above architecture to answer analysis queries that are common in process mining. Specifically, we show how to construct a frequency and time-annotated Directly-Follows Graph (DFG), which is a starting point for process discovery algorithms and performance analysis. While keeping the computed DFG private, we are revealing only the output of performance analysis queries such as finding the top-k bottlenecks (i.e. activities with longer cycle time) or the top-k most frequent hand-offs. We implement our proposed architecture using the Sharemind platform [46]. In order to tackle scalability issues that would be imposed by a naive implementation, we employ vectorization of event logs and propose a divide-and-conquer scheme for parallel processing of sub-logs. We test the effectiveness of these optimizations in experiments with real-world event logs.

This chapter is structured as follows. Sect. 7.1 presents the attack model. Sect. 7.2 introduces our architecture for privacy-preserving interorganizational process mining, along with the optimizations needed for efficient implementation. Sect. 7.3 presents an open source tool to enable privacy-preserving interorganizational process mining. An experimental evaluation is presented in Sect. 7.4, before Sect. 7.5 concludes the chapter. This chapter is derived from [17] and [18] and contains sentences or fragments of sentences from these prior publications.

7.1. Attack Model

In this chapter, we use three-party multi-party computation protocol. This protocol is secured against honest-but-curios adversaries. Therefore, as long as the parties are following the protocol honestly and do not collude, none of them will learn more than the size of the data. We assume that input parties are sharing with each

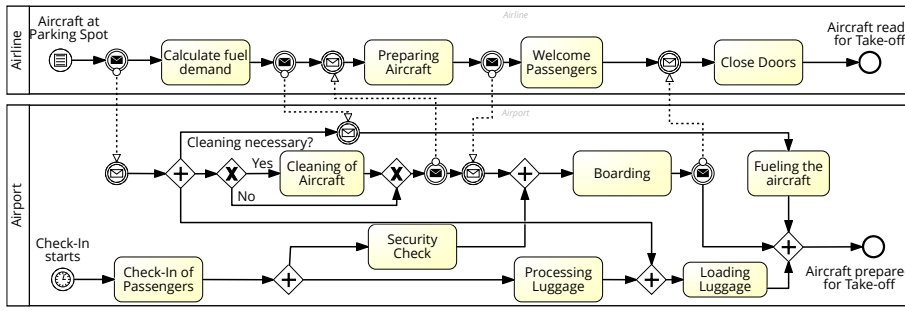


Figure 11. Aircraft ground handling process.

other the number of activities and the maximum trace length in their event logs. Even with encrypted data, contextual knowledge might lead to leakage of some data [68]. Specifically:

- An adversarial party might learn the shortest or the longest trace and with the domain experience, they can reveal the actual activities.
- A leakage might happen due to frequent pattern mining or any access pattern attacks [68].

7.2. Approach

This section introduces our techniques for process mining based on secure multi-party computation. We first clarify our model for interorganizational process mining, including the required input data and the obtained analysis results. We then introduce our architecture for realizing the respective analysis using secure multi-party computation. Furthermore, we elaborate on vectorization and parallelization to improve the efficiency of our approach.

7.2.1. Model for Inter-organizational Process Mining

For an interorganizational business process, an event log that records the process execution from start to end is commonly not available. Rather, different parties record sub-logs, built of events that denote activity executions at the respective party. To keep the notation concise, we consider a setting in which two parties, I_a and I_b, execute an interorganizational process. An example of such process is the process for ground handling of an aircraft, as illustrated in Figure Figure 11. This process involves two parties: the airline and the ground handler (called "airport" in the model). Then, each of the two parties records an event log, denoted by L_a and L_b. Each of these logs is the projection of L on the events that denote activity executions at the respective parties I_a and I_b. We assume that each activity can only be executed by one of the parties, so that this projection is defined unambiguously.

For the above setting, we consider the scenario that the parties I_a and I_b want to answer some analysis queries Q over the interorganizational event log L, yet *without* sharing their logs L_a and L_b with each other. More specifically, we focus on

analysis queries that can be answered on a frequency or time-annotated DFG of the interorganizational process. The basic DFG captures the frequencies with which the executions of two activities have been observed to directly follow each other in a trace. Moreover, we consider temporal annotations of the directly-follows dependencies in terms of time between the respective activity executions. Queries over the frequency and time-annotated DFGs allow us to analyze the main paths of the process, the rarely executed paths, as well as the activities that most contribute to delays in a process. This chapter provides two approaches for two different use cases, the first one assumes that only query answers are to be revealed, whereas the actual DFG shall be kept private. This setting allows the organizations to locate some of the bottlenecks without a deep analysis of the entire DFG. The second approach releases a differentially-private version of the DFG. This setting allows the process analysts to access the DFG while mitigating re-identification attacks and without guessing the actual values in the original log.

Formally, the time-annotated DFG is captured by an $|A| \times |A|$ matrix, where A is the set of all possible activities of the process. Each cell contains a tuple (c, Δ). The counter c represents the frequency with which a directly-follows dependency has been observed in L, i.e., for the cell (a_1, a_2) it is the number of times that two events $e_1 = (i_1, a_1, ts_1)$ and $e_2 = (i_2, a_2, ts_2)$ follow each other directly in some trace (i.e., $i_1 = i_2$) of L. Also, Δ is the total sum of the time passed by between all occurrences of the respective events, i.e., $ts_2 - ts_1$ for the above events.

In interorganizational process mining, the above time-annotated DFG cannot be computed directly, as this would require the parties to share their sub-logs.

7.2.2. MPC Architecture for Process Mining

To enable interorganizational process mining *without* requiring parties to share their event logs with each other, we propose an architecture based on secure multi-party computation (MPC). As outlined in Figure Figure 12, we rely on a platform for MPC (in our case Sharemind [46]) that takes the event logs of the participating parties, i.e., L_a and L_b, as secret-shared input. Inside the MPC platform, the respective data is processed in a privacy-preserving way in order to answer analysis queries over the time-annotated DFG computed from that data. In Figure Figure 13, we present a running example of the processing steps of the system.

Below, we summarize the functionality embodied in the proposed MPC platform for interorganizational process mining.

Preprocessing. Each party performs the preparation of its log at its own site. The parties share the number of unique activities and the maximum number of events per trace. In Figure 13(a), we show an example with two traces. In the preprocessing step, all traces are padded to the same length, as illustrated with the blue event in Figure 13(a). The activities are transformed into a one-hot encoding that is used for masking at the DFG calculation step, as will be explained later. The logs are sorted by traces.

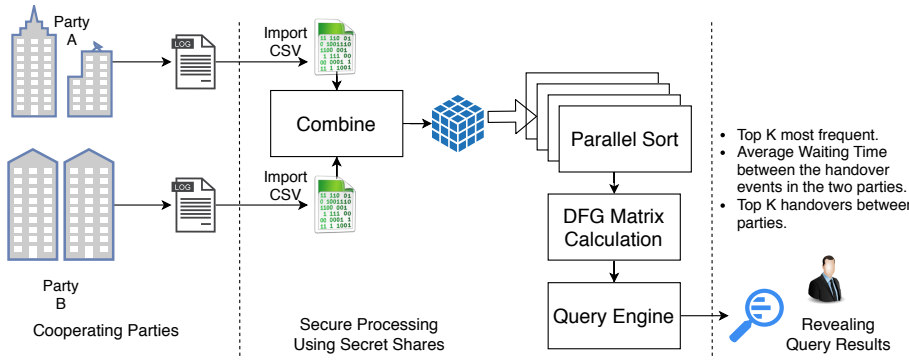
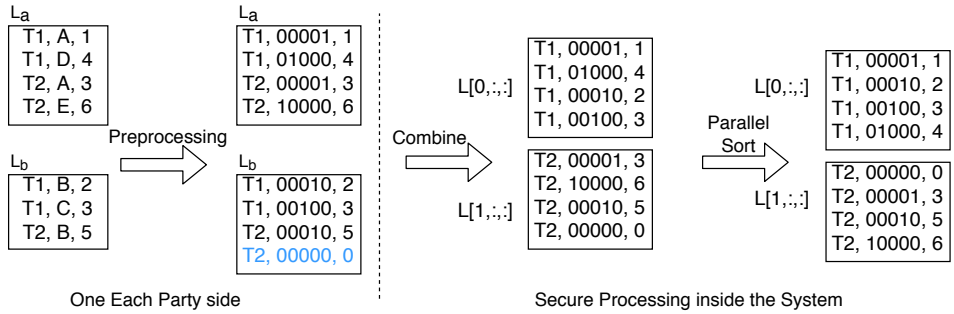


Figure 12. Overview of the proposed approach

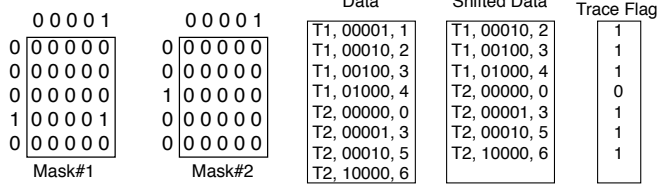
Combination. The parties upload their event logs L_a and L_b to the MPC platform in a secret-shared manner. That is, the values (i,a,ts) of each event (encoded as integers) are split into shares, which do not provide any information on the original values and are stored at different nodes of the platform. This way, each party can only see the total number of records uploaded by each party, but not the particular data. Subsequently, the logs are unified, creating a single log of events L. The combination is performed in a manner to divide the logs into processing chunks. As long as we are making the number of events per trace is fixed (using padding as explained in the previous step), that is possible by dividing the index by the number of traces for each event and assigning data from the same trace to the same chunk. In Figure 13(a), the system processes one trace with its own chunk.

Sorting. To calculate the annotated DFG, we have to determine which events follow each other in a trace by grouping the events by their trace identifier and ordering them by their timestamp. Since the trace identifier is secret-shared, we cannot group events directly. Instead, we use a privacy-preserving quick sort algorithm [114] as implemented in Sharemind to sort the events by their trace identifier. Applying the same algorithm also to the secret-shared timestamps ensures that the events of the same trace follow each other in the order of their timestamps, which is illustrated as the last step in Figure 13(a).

DFG matrix calculation. Next, we construct the DFG matrix inside the MPC platform, keeping it secret. Since the information on the activity of an event is secret-shared, we cannot simply process the events of traces sequentially, as the matrix cell to update would not be known. Hence, we adopt a one-hot encoding for activities, so that each possible activity is represented by a binary vector of length $|A|$. To mask the actual number of possible activities, the set over which the vector is defined may further include some padding, i.e., the vector length can be chosen to be larger than $|A|$. Now, if we compute the *outer product* of such vectors for activities a_1 and a_2, we get a mask matrix M such that $M[a_1,a_2] = 1$, while all other entries are 0. An example of such masks is given in Figure 13(b). The first mask represents the directly-follows dependency from activity A to B of our running example. The second mask encodes the directly-follows dependency



(a) Illustration of the preprocessing, combine and parallel sort steps



	A	B	C	D	E
A	0	0	0	0	0
B	2	0	0	0	0
C	0	1	0	0	0
D	0	0	1	0	0
E	0	1	0	0	0

(b) Example of two masks (c) DFG Calculation using shift and a flag (d) The DFG matrix with counts

Figure 13. Example of two event logs and their processing steps inside the system

from activity A to C. For all sequential pairs of events in the sorted log, we sum up these matrices to get the frequency count c of the directly-follows dependency for (a_1, a_2). Multiplying M by the duration between two events further enables us to derive the total sum duration passed, i.e., Δ, of the directly-follows-relation. The duration operation is performed between every two consecutive events of the same trace. We can perform the duration calculation by using an element-wise vector subtraction by duplicating the event log and then shifting its events by one as in Figure 13(c) . Technically, the outer product is a function that is realized as a protocol over secret-shared data in Sharemind, and its runtime complexity is linear in $|A|$ [115].

However, the above approach could mix up events of different traces. We therefore also compute a flag b that is 1, if the trace identifiers of two events are equivalent, and 0 otherwise, which is illustrated as the "Same Trace Flag" column in Figure 13(c). Then, we multiply the mask matrix M by b, so that the values of M are ignored, if $b = 0$. Again, the functionality for comparison and multiplication can be traced back to predefined protocols in Sharemind. We show the DFG matrix with counts of our running example in Figure 13(d).

Algorithm algorithm 4 summarizes the computation of the annotated DFG from the sorted, combined log L, where $[\![\cdot]\!]$ denotes a secret-shared data value. First, the approach computes the flags that determines whether each to consecutive activities belong to the same case or not. Then, the approach performs the outer product to calculate the frequency of each directly-follows relation (each mask). After that,

the approach adds the masks together to construct the frequency-annotated DFG. Lastly, the approach performs the outer product to calculate the waiting time, and sums the total waiting time per directly-follows relation. At the end, the approach returns both the frequency and time-annotated DFGs.

Query answering.. A query Q defines a subset S of the annotated DFG, which is generated by the MPC platform and revealed to the participating parties. Through sharing the S solely, but not the complete annotated DFG, we are able to limit the amount of information each party can learn about the process. As an example query, consider the query to derive the average waiting time between the handover events between the two parties. Based on the secret-shared DFG, the respective activities may be identified through grouping and sorting the events, similar to the procedure outlined above, which is again based on the predefined protocols of an MPC platform such as Sharemind.

Algorithm 4: Calculating the combined, annotated DFG($[\![L]\!]$)

Input: $[\![L]\!]$: The sorted, combined event log of length n.
Output: Annotated DFG comprising a count matrix $[\![G]\!]$
Initialize $[\![G]\!] = 0$, $[\![W]\!] = 0$
for $j \in \{\, 1, \ldots, n-1 \,\}$ **do**
  $[\![b]\!] \leftarrow ([\![L[j-1].i]\!] = [\![L[j].i]\!])$; ▷ compute the flag for traces.
  $[\![M]\!] \leftarrow [\![b]\!] \cdot ([\![L[j-1].a]\!] \otimes [\![L[j].a]\!])$; ▷ compute the outer product.
  $[\![G]\!] \leftarrow [\![G]\!] + [\![M]\!]$; ▷ incorporate the current dependency.
  $[\![W]\!] \leftarrow [\![W]\!] + [\![M]\!] \cdot ([\![L[j].ts]\!] - [\![L[j-1].ts]\!])$; ▷ Incorporate the time lag.
end
return $[\![G]\!]$, $[\![W]\!]$

7.2.3. Performance Optimizations

Interorganizational process mining using the above general architecture might suffer from scalability issues. The reason is that privacy-preserving computation through protocols over secret-shared data is inevitably less efficient than plain computation. Hence, even for functions that have a generally low runtime complexity ($\mathcal{O}(n)$ for the combination, $\mathcal{O}(n\log(n))$ for the sorting, $\mathcal{O}(nm^2)$ for the calculation of the annotated DFG, where n is the log length and m is the number of activities), there is a non-negligible overhead induced by MPC. For instance, a naive realization of the quick sort algorithm to sort events would require $\mathcal{O}(n\log(n))$ rounds of communication between the nodes and $\mathcal{O}(n\log(n))$ value comparisons per round [114]. We therefore consider two angles to improve the efficiency of the analysis, namely vectorization and parallelization.

Vectorization. A computation that adopts a single-instruction multiple-data (SIMD) approach is highly recommended in MPC applications. Since MPC assumes continuous interaction between distributed nodes, the number of communi-

cation rounds shall be reduced as much as possible. For instance, while computing n multiplications sequentially would result in n rounds of communication, one may alternatively multiply element-wise two vectors of length n, for which one round of network communication is sufficient. Sharemind offers efficient protocols for such vector-based functions [115].

Parallelization. Further runtime improvements are obtained by parallelizing the algorithm itself. Again, our goal is to reduce the number of rounds of communication among the nodes of the MPC platform. We, therefore, split the input data into *chunks*, such that all chunks can be processed independently of each other. In our scenario, this is done by grouping the party logs by trace, or by a group of traces, generating an annotated DFG per group, and finally integrating the different DFGs. Since events of the same trace will never occur in different chunks, instead of sorting one log of length n, we will need to sort c chunks of length n/c each. Since the communication complexity of a privacy-preserving quick sort is $O(n \cdot \log n)$ [114], this improves efficiency.

The above approach raises the question of determining the size of the chunks. Separating each trace reveals the total number of events of that trace provided by a party, which may be critical from a privacy perspective. On the other hand, a small chunk size reduces the overhead of sorting. This leads to a trade-off between runtime performance and privacy considerations.

However, in our current implementation, all chunks must have the same length, as Sharemind allows parallel sorting only for equal-length vectors. Therefore, we apply padding to the traces in the log, adding dummy events (for which an empty vector in the one-hot encoding represents the activity so that the events are ignored for the DFG calculation) until the number of events of the longest trace is reached. Such padding may be employed locally, by each party, and also has the benefit that the length of individual traces is not revealed.

7.3. Software Implementation

In this section, we present Shareprom [1], a tool that we developed to provide users access to the approach proposed in this chapter. Shareprom allows independent cooperating organizations to control what is disclosed from their event logs. We assume that the parties have agreed to release their process map. In other words, the analyst of the released process map will not learn anything about the independent organizations other than the release output. Shareprom applies differential privacy to the resulting DFG to provide an extra layer of privacy. The analyst will neither have access to specific traces nor to any information associated with these traces. And naturally, the data providers will not able to learn anything about each other's private data.

[1] Corresponding to [18].

7.3.1. Overview of Shareprom

Shareprom uses the three-party MPC protocol set of Sharemind, which is secure against honest-but-curious adversaries. This means that as long as the parties are following the protocols honestly and do not collude, none of them will learn more than the size of the data. Parties in a typical secure MPC deployment can be: *input parties*, *computation parties*, or *output parties* [116]. The sets of input and computing parties may intersect. In this paper, we assume a simplified scenario, where the three computing parties themselves provide the inputs and receive the outputs. Two of the parties contribute the input data, and the third party is the analysis firm that receives the final output.

We assume that input parties share with each other the number of activities and the maximum trace length in their event logs. Also, we assume that the case identifiers are the same across the input parties. This is needed to conduct preprocessing that reduces the amount of information that might be learned from the size of data. Even with encrypted data, contextual knowledge might lead to leakage of some information. An adversarial party might learn the shortest or the longest trace, and using the domain knowledge, they could reveal the actual activities. For such a case, we apply padding to the logs on the client side of each input party, so the resulting log has all the traces with the same length, which is set to the maximum trace length. Parties can hide the maximum trace length by adding extra padding. The logs are uploaded to computation servers in a secret-shared manner, and the MPC protocol performs computation without any intermediate declassification. As such, during the computation, the parties do not learn anything in addition to the sizes of padded logs. This also excludes any attacks related to access patterns, like frequent pattern mining, which would be possible, if the events that are equal to each other, are leaked.

Figure Figure 14 gives an overview of the system components. Below, we summarize the functionality of each component of Shareprom. More information about the system components can be found in Sec. 7.2.

Preprocessing. Each party of the cooperating organization uses Shareprom clients to import the XES file of their event log. Shareprom then performs preprocessing of the event log at its organization site. The parties exchange the maximum trace length and the number of unique activities of their entire log. All the traces need to be padded to have the same length as the longest trace. The activities are mapped to a one-hot encoding format, independently, on each input party. Also, a sort step of the logs by traces is performed. The two latter steps are needed as a preparation for the subsequent secure MPC protocol, which requires the data to be available in a specific format.

Mapping Event Log to Secret-shares. Each party uses their Shareprom client to map their event logs into secret-shares. Each client pushes its secret-shares to the Sharemind servers. Secret-shares do not reveal any information about the event log. Until this point, the analysis firm, as one of the computation parties, has only

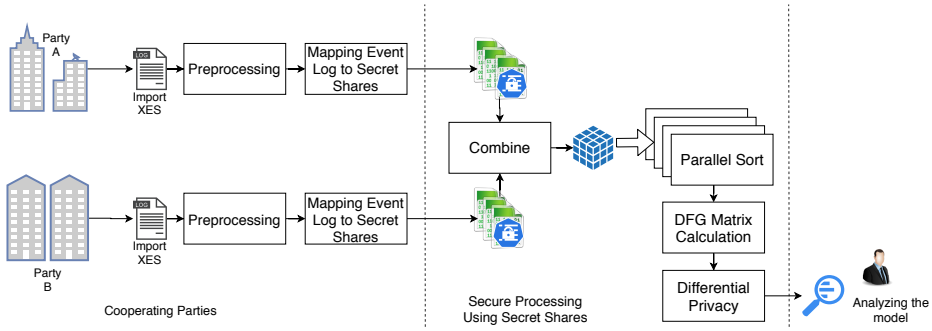


Figure 14. Overview of Shareprom

received a single share of each input, which alone looks like a random value.

DFG Matrix Calculation. Shareprom runs the Sharemind MPC protocol to construct the DFG matrix from uploaded secret-shared data. An algorithm based on a one-hot encoding technique allows us to ensure that the computation does not reveal any information to the parties, including the access pattern (e.g. which log entries follow each other). The details of this algorithm are presented in Sec. 7.2. The resulting DFG takes the form of a matrix with secret-shared entries, where each entry corresponds to a transition between two activities, containing the frequency count and the total execution time for that transition.

Differential Privacy. The DFG itself may reveal some information about the parties' inputs. We consider the frequency and time difference between each pair of activities in the DFG as a separate query. Hence, before disclosing the final result to the analyst party, we enhance it with differential privacy by injecting Laplacian noise to the frequency and time difference of each pair of activities. The added noise conceals the order of activities and their execution times. We consider understanding the trade-offs between the amount of added noise and the utility of the output as a direction for future work.

7.3.2. Tool Packaging and Maturity

Shareprom is developed in Python 3.7 and SecreC – a dialect of the C language supported by the Sharemind platform [47]. For ease of use, Shareprom comes with a desktop client application. The desktop client allows a user to import an event log (in XES format). The desktop client connects with its respective Shareprom server. In a typical configuration, three Sharemind servers are interconnected, e.g. two input providers and a computation node – the latter representing for example an analysis firm. The analysis firm can view the output (with differential privacy noise). An analyst at this firm can then analyze the output and share the findings (or the whole output) with the input providers, depending on the intended use case. The source code, installation steps, dependencies, screencast and example event logs could be found at `https://github.com/Elkoumy/shareprom/releases/tag/v0.2`.

7.4. Evaluation

We implemented the proposed approach on top of the Sharemind multi-party computation platform.[2] The source code of our implementation is available at https://github.com/Elkoumy/shareprom. The implementation is written using the SecreC programming language supported by Sharemind.

Using this implementation, we conducted feasibility and scalability experiments, specifically to address the following questions:

Q1: How do the characteristics of the input event logs influence the performance of the secure multi-party computation of the DFG?

Q2: What is the effect of increasing the number of parallel chunks on the performance of the multi-party computation of the DFG?

7.4.1. Event Logs

The proposed approach is designed to compute the DFG of an interorganizational process where the event log is distributed across multiple parties, and each party is responsible for executing a subset of the activities (i.e. *event types*) of the process. We are not aware of publicly available real-life event logs with this property. We identified a collection of synthetic interorganizational business process event logs [117]. However, these logs are too small to allow us to conduct performance experiments (a few dozen traces per log). On the other hand, there is a collection of real-life event logs of intra-organizational processes comprising logs of varying sizes and characteristics [3]. From this collection, we selected three logs with different size and complexity (cf. Table 17):

BPIC 2013 This event log captures incident and problem management process at an IT department of a car production company.

Credit Requirement This event log comes from a process for background checking for the purpose of granting credit at a Dutch bank. It has a simple control-flow structure: All traces follow the same sequence of activities.

Traffic Fines This event log comes from a process to collect payment of fines from traffic law violations at a local police office in Italy.

Table 17. Event Logs for Evaluation

Event Log	# Events	# Cases	# Activities	# Events in Case		
				Avg	Max	Min
BPIC 2013	6,660	1,432	6	4.478	35	1
Credit Requirement	50,525	10,034	8	15	15	15
Traffic Fines	561,470	150,370	11	3.73	20	2

[2] https://sharemind-sdk.github.io

[3] https://data.4tu.nl/repository/collection:event_logs_real

To simulate an interorganizational setting, we use a round-robin approach to assign each event type (activity) in the log to one of two parties. Hence, each party executes half of the event types.

7.4.2. Experimental Setup

To answer the above questions, we use the following performance measures:

- **Runtime.** We define runtime as the amount of time needed to transform the event logs of the two parties securely into an annotated DFG. We also report the throughput, the number of events processed by the system per second, to provide a complementary perspective.
- **Communication Overhead.** We define the communication overhead as the amount of data transferred between the computing parties during the multi-party computation. We measure this overhead as the volume of the data sent and received. The communication overhead gives insights into how much the performance of the multi-party computation would degrade if the computing nodes of the parties were distributed across a wide-area network.

We performed five runs per event log per experiment. We report the average maximum values for latency and the average value for both throughput and communication overhead, across the five runs. We used *Nethogs*[4] to measure the communication overhead, and we report the average value per compute node. The experiments were run in an environment with three physical servers as compute nodes with Sharemind installed on them. Each server has an AMD Processor 6276 and 192 GB RAM. The servers are connected using a 1 GB Ethernet switch.

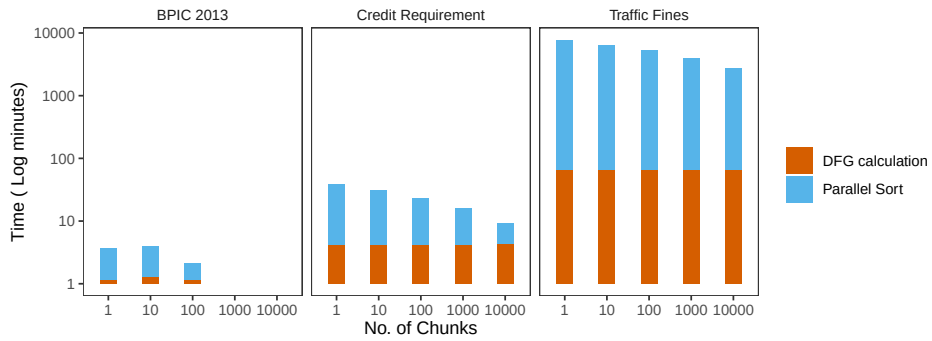
The experiments focus on the time needed to construct the annotated DFG, since it is the most sophisticated and time-consuming portion of the proposed analysis pipeline, due to the communication required between the compute nodes. Once the annotated DFG is available, stored in a secret-shared manner, the calculation of the actual queries has a lower complexity.

7.4.3. Results

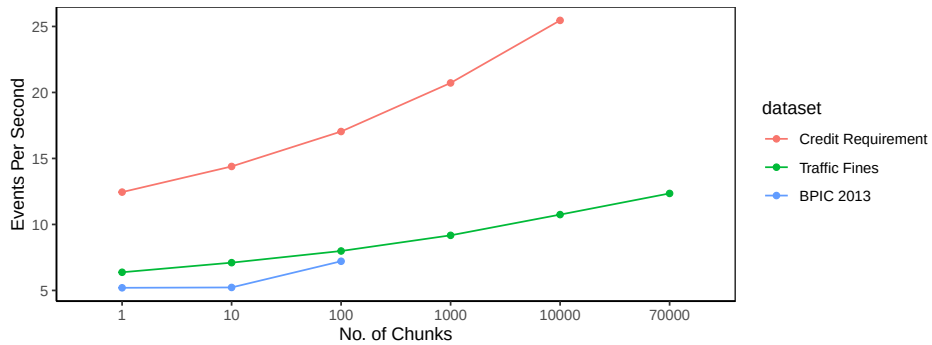
Runtime Experiment.. In Figure 15(a), we illustrate the observed execution time when varying the number of chunks used in the parallelization. We plot a bar for each chunk size. Each bar represents the runtime of the parallel sort in blue and the run time of the DFG calculation in orange. From Figure 15(a), we conclude that the runtime decreases with an increasing number of chunks, due to the parallel sorting of chunks. We also note that the runtime for the DFG calculation stays constant. In Figure 15(b), we report the number of processed events per second when varying the number of chunks. We find a consistent improvement for the throughput across all event logs.

Regarding Q1, we summarize that the proportion of runtime between sorting and DFG calculation differs based on the event log characteristics. For the log with the

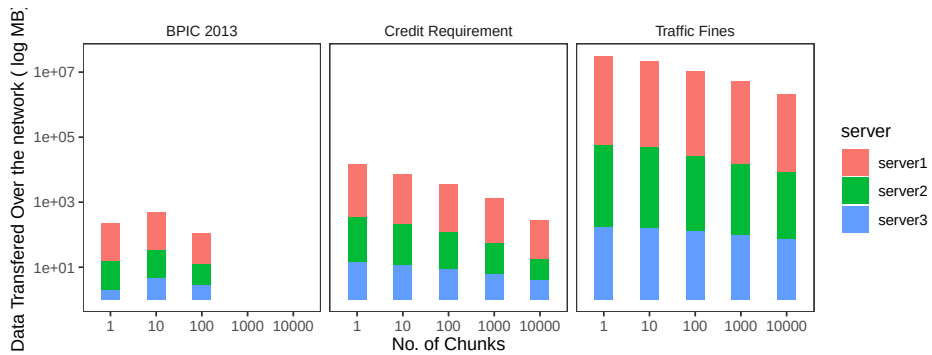
[4] https://github.com/raboof/nethogs



(a) Runtime Experiment: Execution Time (Log) vs no. of Chunks.



(b) Throughput Experiment: Events per Second vs no. of Chunks.



(c) Communication Overhead Experiment: Data Transferred (Log) vs no. of Chunks.

Figure 15. Example of two event logs and their processing steps inside the system

largest number of event types, the DFG calculation makes up the most substantial proportion of the total runtime. In contrast, the proportion is significantly lower for the logs with a smaller number of event types. A possible explanation for this finding is the increasing size of the vectors required to represent each activity due to our bit-vector representation. Such increase results in more computational heavy calculations. Regarding Q2, we conclude that the runtime decreases for event logs with an increasing number of chunks.

Communication Overhead.. In Figure 15(c), we present the amount of data transferred to each server, again also varying the number of chunks. We observe that the communication overhead decreases with an increase in the number of chunks. These findings confirm our earlier findings regarding Q2. In summary, a higher number of chunks leads to improved performance across all three measures.

Threats to validity.. The evaluation reported above has two limitations. First, the event logs used in the evaluation, while coming from real-life systems, are intra-organizational event logs, which we have split into separate logs to simulate an interorganizational setting. It is possible that these logs do not capture the communication patterns found in interorganizational processes. Second, the number of event logs is reduced, which limits the generalizability of the conclusions. The results suggest that the proposed technique can handle small-to-medium-sized logs, with relatively short traces, but there may be other characteristics of event logs that affect the performance of the proposed approach.

7.5. Summary

This chapter introduced a framework for interorganizational process mining based on secure multi-party computation. The framework enables two or more parties to perform basic process mining operations over the partial logs of an interorganizational process held by each party, without any information being shared besides: (i) the output of the queries that the parties opt to disclose; and (ii) three high-level log statistics: the number of traces per log, the number of event types, and the maximum trace length. The chapter specifically focuses on the computation of the DFG, annotated with frequency and temporal information. This is a basic structure used by process mining tools to perform various operations, including automated process discovery and various performance analysis queries (e.g. top-k bottlenecks and least-frequent and most-frequent flow dependencies).

To mitigate the high-performance overhead commonly observed for secure multi-party computation, we introduced two optimizations over the basic DFG computation algorithm: one based on vectorization of the event log and the other based on a divide-and-conquer strategy, where the log is processed in chunks.

An evaluation using real world event logs shows that with these optimizations, it is possible to compute the DFG of real-life logs with execution times that make this technique usable in practice. The divide-and-conquer approach provides opportunities to scale up the proposed technique by using a map-reduce execution-

style, however not to a sufficient level to enable interactive process mining (which requires execution times in the order of seconds). Also, the approach is not able to handle logs with thousands of traces.

8. CONCLUSION

8.1. Summary of Contributions

This thesis addressed the question of how to anonymize event logs and perform process mining while preventing or controlling the release of personal information of the individuals for whom the process is being executed. Specifically, the thesis addressed two overarching questions:

RQ1 Given an event log L, wherein each trace contains private information about an individual (e.g., a customer), and given a privacy risk threshold, how to generate an anonymized event log L' that provides a differential privacy guarantee to each individual represented by a trace in the log, w.r.t. the given threshold?

RQ2 How to enable process mining for interorganizational business processes without requiring the involved parties to share their private event logs or to trust a third party?

Any anonymization technique needs to strike a balance between controlling privacy risks and allowing the user (in our case, an analyst) to draw useful conclusions from the anonymized event logs. In the context of process mining, it is important to ensure that the anonymized event log still represents the same sequences of activities (with more or less the same frequencies) and that the timestamps associated to the events in the traces of a log resemble those in the original log. Concurrently, the anonymization mechanism should mitigate the situation where there exists a predicate that uniquely identifies a case in the log (PSO-secure). This remark leads us to associate the following Utility Requirements to research question RQ1:

- **UR1a**: The anonymized event log must have the same set of case variants as the original log.
- **UR1b**: The anonymized event log must not introduce new case variants to the original log.
- **UR2**: The difference between the real and the anonymized time values is minimal given a privacy risk metric.

To address RQ1 under UR1a and UR2, we proposed a concept of differentially private event log and a mechanism to compute such logs (cf. Contribution 1). A differentially private event log limits the increase in the probability that an attacker may learn a suffix of an individual's trace given a prefix (or vice-versa), or the timestamp of an activity in an individual's trace. To this end, we inject differentially private noise by oversampling the traces in the log. This approach neither suppresses nor adds case variants, and hence fulfills UR1a. To minimize the difference between the real and the anonymized time values (UR2), we quantify ε based on a technique that finds the maximum ε (minimum noise) that keeps the guessing advantage below δ.

The proposed approach is implemented as an open source tool. Also, the

approach has been evaluated using 14 real-life event logs that are publicly available. This approach fits the anonymization of structured event log. A limitation of this proposed method is that it anonymizes the relative time of each event with respect to the start of the case, but it does not anonymize the start times of the cases. A second limitation is that the input log is assumed to have only three columns: case ID, activity label, and timestamp. Real-world event logs usually contain other columns, e.g., resources.

The approach presented in Contribution 1 introduces high levels of noise in the presence of unique traces or temporal outliers. To address this limitation, in Contribution 2, we relaxed the utility requirement to make sure that the anonymized event log must not introduce new case variants (UR1b) to the original log. Contribution 2 presented an approach that applies both over- and undersampling, as opposed to only oversampling; and filtering out high-risk cases. In that way, the proposed approach does not introduce high levels of noise with unstructured event logs. The approach presented in Contribution 2 fits the anonymization of unstructured event log.

The proposed approach has been empirically evaluated and a comparison with the state-of-the-art has been conducted. The empirical findings show that the proposed approach is a step toward anonymizing event logs while preserving the utility of the process mining analysis. The proposed approach outperforms the state-of-the-art in terms of Jaccard distance and earth movers' distance, and generalization to all fourteen real-life event logs selected in the evaluation. Furthermore, the approach can process large-sized event logs in practical memory size (32 GB). A limitation of the proposed approach is that it do not handle resource and other columns' anonymization.

Contribution 3 proposed a method that enhances the noise injection of the differentially-private mechanism of event logs using privacy amplification. While previous proposals rely purely on noise injection, the approach proposed in Contribution 3 additionally employs subsampling to achieve stronger privacy guarantees with the same level of utility loss, or conversely, less utility loss for the same privacy guarantee (privacy amplification). The approach presented in Contribution 3 fits the anonymization of structured event log when the number of rare case variants is small.

The proposed approach is implemented as an open source tool called Libra. Libra has been evaluated using 8 real-life event logs against the state-of-the-art. The empirical evaluation shows that the privacy amplification effect leads to significant reductions of utility loss, particularly when it comes to anonymizing the frequency of distribution of case variants in a log (i.e. control-flow anonymization) and to a lesser extent when it comes to anonymization of event timestamps.

A limitation of Libra is that it is not suitable for event logs with a high proportion of infrequent trace variants. That is because Libra depends on subsampling. If we have many unique trace variants, no matter the size of the subsample, the unique traces will appear in the subsampled log. In such use cases, Libra simply filters out

most of the traces and may lead to empty outputs. Also, Libra returns empty event logs when all the traces in the event log are unique traces.

To address RQ2, Contribution 4 introduced a framework for interorganizational process mining based on secure multi-party computation. The framework enables two or more parties to perform basic process mining operations over the partial logs of an interorganizational process held by each party, without any information being shared besides: (i) the output of the queries that the parties opt to disclose; and (ii) three high-level log statistics: the number of traces per log, the number of event types, and the maximum trace length. Contribution 4 specifically focused on the computation of the DFG, annotated with frequency and temporal information. This is a basic structure used by process mining tools to perform various operations, including automated process discovery and various performance analysis queries (e.g. top-k bottlenecks and least-frequent and most-frequent flow dependencies).

The proposed approach is implemented as an open-source tool called Shareprom. Shareprom mitigates the high-performance overhead commonly observed for secure multi-party computation, we introduced two optimizations over the basic DFG computation algorithm: one based on vectorization of the event log and the other based on a divide-and-conquer strategy, where the log is processed in chunks. An evaluation using real world event logs shows that with these optimizations, it is possible to compute the DFG of real-life logs with execution times that make this technique usable in practice. A limitation of Shareprom is that it is not able to handle logs with thousands of traces. Another limitation of Shareprom is that it focuses on computing DFGs, and does not support other process mining operations such as conformance checking, variant analysis, etc.

8.2. Future work

The contributions presented in this thesis open up future research directions, as described in the following.

Anonymizing Resources and Other Attributes. As explained above, Contributions 1, 2, and 3 share the limitation that they all only work from the workflow perspective. Real-world event logs contain other columns, e.g., resources. To address this limitation, we need to extend the log representation, e.g., via multidimensional data structures instead of DAFSAs. Also, we aim to build a differentially private mechanism with different, smaller building blocks. Each building block achieves differential privacy guarantees for each perspective separately. Combining these building blocks is possible using the composition theorem.

Anonymizing Event Logs with Large Proportion of Unique Case Variants. As we elaborated above, a limitation of Libra is that it is not suitable for event logs with a high proportion of infrequent trace variants. In such use cases, Libra simply filters out most of the traces and may lead to empty outputs. A possible solution to this limitation is using methods to preserve the unique trace variants in the log and minimize the utility loss. This can be achieved by clustering multiple similar

case variants into a single cluster and then replacing the case variants with the centroid of the cluster, which would then have a frequency equal to the sum of the frequencies of the trace variants in the cluster. One way to find the centroid of the clusters is using summarization techniques. Investigating these two methods is a possible future research avenue.

Usability Evaluation of Event Log Anonymization Techniques. In privacy-preserving process mining techniques, particularly anonymization techniques, it is assumed that the event log owner clearly understands the provided privacy guarantees and parameters. In this thesis, we postulated that by introducing the guessing advantage parameter. However, this postulate has not been tested in practice. A possible future research avenue is to conduct an evaluation with users and a controlled environment to study the ability of users to understand the privacy/utility metrics that these techniques provide (interpretability of metrics). Furthermore, the study should cover the usefulness (utility) of the anonymized logs to perform the analysis by process analysts.

Scalable and High-Performance Distributed Privacy. As we explained above, one limitation of Shareprom, is that it is not suitable for large scale event logs (cf. Chapter 7). The main reason for that is that the secure multi-party computation protocols require a high communication overhead between the nodes. This prevents the protocols to scale with large-scale event logs. A possible solution to this problem is to use low latency but more specific protocols, e.g., set intersection protocols.

BIBLIOGRAPHY

[1] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. Springer, 2013.

[2] E. G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) 2016," *OJ L*, vol. 119, no. 1, 2016.

[3] "Article 29 data protection working party, opinion 05/2014 on anonymisation techniques." Accessed: 19 Nov 2020.

[4] A. Cohen and K. Nissim, "Towards formalizing the gdpr's notion of singling out," *Proc. Natl. Acad. Sci. USA*, vol. 117, no. 15, pp. 8344–8352, 2020.

[5] I. Wagner and D. Eckhoff, "Technical privacy metrics: A systematic survey," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 57:1–57:38, 2018.

[6] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy.," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, 2014.

[7] J. Lee and C. Clifton, "How much is enough? choosing ε for differential privacy," in *Proc. ISC.*, pp. 325–340, Springer, 2011.

[8] C. Dwork, N. Kohli, and D. Mulligan, "Differential privacy in practice: Expose your epsilons!," *J. Priv. Confidentiality*, vol. 9, no. 2, 2019.

[9] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 4, pp. 686–705, 2019.

[10] H. van der Aa, H. Leopold, and H. A. Reijers, "Efficient process conformance checking on the basis of uncertain event-to-activity mappings," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 5, pp. 927–940, 2020.

[11] M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[12] M. Rafiei, M. Wagner, and W. M. P. van der Aalst, "Tlkc-privacy model for process mining," in *Research Challenges in Information Science - 14th International Conference, RCIS 2020, Limassol, Cyprus, September 23-25, 2020, Proceedings* (F. Dalpiaz, J. Zdravkovic, and P. Loucopoulos, eds.), vol. 385 of *Lecture Notes in Business Information Processing*, pp. 398–416, Springer, 2020.

[13] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004.

[14] J. Daciuk, S. Mihov, B. W. Watson, and R. Watson, "Incremental construction of minimal acyclic finite state automata," *Comput. Linguistics*, vol. 26, no. 1, pp. 3–16, 2000.

[15] G. Elkoumy, A. Pankova, and M. Dumas, "Mine me but don't single me out: Differentially private event logs for process mining," in *3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021* (C. D. Ciccio, C. D. Francescomarino, and P. Soffer, eds.), pp. 80–87, IEEE, 2021.

[16] G. Elkoumy and M. Dumas, "Libra: High-utility anonymization of event logs for process mining via subsampling," 2022.

[17] G. Elkoumy, S. A. Fahrenkrog-Petersen, M. Dumas, P. Laud, A. Pankova, and M. Weidlich, "Secure multi-party computation for inter-organizational process mining," in *Enterprise, Business-Process and Information Systems Modeling - 21st International Conference, BPMDS 2020, 25th International Conference, EMMSAD 2020, Held at CAiSE 2020, Grenoble, France, June 8-9, 2020, Proceedings* (S. Nurcan, I. Reinhartz-Berger, P. Soffer, and J. Zdravkovic, eds.), vol. 387 of *Lecture Notes in Business Information Processing*, pp. 166–181, Springer, 2020.

[18] G. Elkoumy, S. A. Fahrenkrog-Petersen, M. Dumas, P. Laud, A. Pankova, and M. Weidlich, "Shareprom: A tool for privacy-preserving inter-organizational process mining," in *BPM Demos*, pp. 72–76, 2020.

[19] A. F. Westin, "Privacy and freedom," *Washington and Lee Law Review*, vol. 25, no. 1, p. 166, 1968.

[20] D. A. Catania, "The universal declaration of human rights and sodomy laws: A federal common law right to privacy for homosexuals based on customary international law," *Am. Crim. L. Rev.*, vol. 31, p. 289, 1993.

[21] G. Greenleaf, "Five years of the APEC privacy framework: Failure or promise?," *Comput. Law Secur. Rev.*, vol. 25, no. 1, pp. 28–43, 2009.

[22] L. B. Harman, C. A. Flite, and K. Bond, "Electronic health records: privacy, confidentiality, and security," *AMA Journal of Ethics*, vol. 14, no. 9, pp. 712–719, 2012.

[23] A. Partington, M. T. Wynn, S. Suriadi, C. Ouyang, and J. Karnon, "Process mining for clinical processes: A comparative analysis of four australian hospitals," *ACM Trans. Manag. Inf. Syst.*, vol. 5, no. 4, pp. 19:1–19:18, 2015.

[24] E. Rojas, J. Munoz-Gama, M. Sepúlveda, and D. Capurro, "Process mining in healthcare: A literature review," *Journal of Biomedical Informatics*, vol. 61, pp. 224–236, 2016.

[25] K. A. Schulz and M. E. Orlowska, "Facilitating cross-organisational workflows with a workflow view approach," *Data & Knowledge Engineering*, vol. 51, no. 1, pp. 109 – 147, 2004.

[26] Q. Zeng, S. X. Sun, H. Duan, C. Liu, and H. Wang, "Cross-organizational collaborative workflow mining from a multi-source log," *Decision Support Systems*, vol. 54, no. 3, pp. 1280–1301, 2013.

[27] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking - Relating Processes and Models*. Springer, 2018.

[28] A. Pika, M. Leyer, M. T. Wynn, C. J. Fidge, A. H. M. ter Hofstede, and W. M. P. van der Aalst, "Mining resource profiles from event logs," *ACM Trans. Manag. Inf. Syst.*, vol. 8, no. 1, pp. 1:1–1:30, 2017.

[29] W. M. P. van der Aalst, *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[30] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Trans. Knowl. Discov. Data*, vol. 13, Mar. 2019.

[31] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd International Conference on Data Engineering*, pp. 106–115, IEEE, 2007.

[32] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[33] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*, pp. 1–19, Springer, 2008.

[34] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings* (S. Halevi and T. Rabin, eds.), vol. 3876 of *Lecture Notes in Computer Science*, pp. 265–284, Springer, 2006.

[35] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially private data publishing and analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, 2017.

[36] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[37] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, Springer, 1999.

[38] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pp. 280–299, 2001.

[39] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 169–178, 2009.

[40] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA* (A. V. Aho, ed.), pp. 218–229, ACM, 1987.

[41] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*, pp. 160–164, IEEE, 1982.

[42] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 162–167, IEEE, 1986.

[43] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[44] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pp. 805–817, 2016.

[45] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases—real-world applications of secure multi-party computation," *The Computer Journal*, vol. 61, no. 12, pp. 1749–1771, 2018.

[46] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*, pp. 192–206, Springer, 2008.

[47] D. Bogdanov, P. Laud, and J. Randmets, "Domain-polymorphic programming of privacy-preserving applications," in *Proceedings of the Ninth Workshop on Programming Languages and Analysis for Security*, p. 53, ACM, 2014.

[48] G. Elkoumy, S. A. Fahrenkrog-Petersen, M. F. Sani, A. Koschmider, F. Mannhardt, S. N. von Voigt, M. Rafiei, and L. von Waldthausen, "Privacy and confidentiality in process mining: Threats and research challenges," *ACM Trans. Manag. Inf. Syst.*, vol. 13, no. 1, pp. 11:1–11:17, 2022.

[49] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," *Annual Review of Statistics and Its Application*, vol. 4, no. 1, pp. 61–84, 2017.

[50] S. N. von Voigt, S. A. Fahrenkrog-Petersen, D. Janssen, A. Koschmider, F. Tschorsch, F. Mannhardt, O. Landsiedel, and M. Weidlich, "Quantifying the re-identification risk of event logs for process mining - empiricial evaluation paper," in *Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings* (S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, eds.), vol. 12127 of *Lecture Notes in Computer Science*, pp. 252–267, Springer, 2020.

[51] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pp. 202–210, ACM, 2003.

[52] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pp. 1322–1333, ACM, 2015.

[53] S. A. Fahrenkrog-Petersen, N. Tax, I. Teinemaa, M. Dumas, M. de Leoni, F. M. Maggi, and M. Weidlich, "Fire now, fire later: Alarm-based systems for prescriptive process monitoring," *arXiv preprint arXiv:1905.09568*, 2019.

[54] F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *International conference on advanced information systems engineering*, pp. 457–472, Springer, 2014.

[55] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 3–18, IEEE Computer Society, 2017.

[56] M. Gharib, J. Mylopoulos, and P. Giorgini, "Copri - a core ontology for privacy requirements engineering," in *Research Challenges in Information Science* (F. Dalpiaz, J. Zdravkovic, and P. Loucopoulos, eds.), (Cham), pp. 472–489, Springer International Publishing, 2020.

[57] M. Gharib, P. Giorgini, and J. Mylopoulos, "Towards an ontology for privacy requirements via a systematic literature review," in *Conceptual Modeling* (H. C. Mayr, G. Guizzardi, H. Ma, and O. Pastor, eds.), (Cham), pp. 193–208, Springer International Publishing, 2017.

[58] S. Dritsas, L. Gymnopoulos, M. Karyda, T. Balopoulos, S. Kokolakis, C. Lambrinoudakis, and S. Katsikas, "A knowledge-based approach to security requirements for e-health applications," *Electronic Journal for E-Commerce Tools and Applications*, pp. 1–24, 2006.

[59] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - A proposal for terminology," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings* (H. Federrath, ed.), vol. 2009 of *Lecture Notes in Computer Science*, pp. 1–9, Springer, 2000.

[60] M. Rafiei and W. M. P. van der Aalst, "Group-based privacy preservation techniques for process mining," *Data Knowl. Eng.*, vol. 134, p. 101908, 2021.

[61] S. A. Fahrenkrog-Petersen, H. van der Aa, and M. Weidlich, "PRETSA: event log sanitization for privacy-aware process discovery," in *International*

Conference on Process Mining, ICPM 2019, Aachen, Germany, June 24-26, 2019, pp. 1–8, IEEE, 2019.

[62] M. Bauer, S. A. Fahrenkrog-Petersen, A. Koschmider, F. Mannhardt, H. van der Aa, and M. Weidlich, "ELPaaS: event log privacy as a service," in *Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019* (B. Depaire, J. D. Smedt, M. Dumas, D. Fahland, A. Kumar, H. Leopold, M. Reichert, S. Rinderle-Ma, S. Schulte, S. Seidel, and W. M. P. van der Aalst, eds.), vol. 2420 of *CEUR Workshop Proceedings*, pp. 159–163, CEUR-WS.org, 2019.

[63] E. Batista and A. Solanas, "A uniformization-based approach to preserve individuals' privacy during process mining analyses," *Peer Peer Netw. Appl.*, vol. 14, no. 3, pp. 1500–1519, 2021.

[64] F. Mannhardt, A. Koschmider, N. Baracaldo, M. Weidlich, and J. Michael, "Privacy-preserving process mining - differential privacy for event logs," *Bus. Inf. Syst. Eng.*, vol. 61, no. 5, pp. 595–614, 2019.

[65] S. A. Fahrenkrog-Petersen, H. van der Aa, and M. Weidlich, "PRIPEL: privacy-preserving event log publishing including contextual information," in *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings* (D. Fahland, C. Ghidini, J. Becker, and M. Dumas, eds.), vol. 12168 of *Lecture Notes in Computer Science*, pp. 111–128, Springer, 2020.

[66] S. A. Fahrenkrog-Petersen, M. Kabierski, F. Rösel, H. van der Aa, and M. Weidlich, "SaCoFa: semantics-aware control-flow anonymization for process mining," in *3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021* (C. D. Ciccio, C. D. Francescomarino, and P. Soffer, eds.), pp. 72–79, IEEE, 2021.

[67] M. Rafiei, L. von Waldthausen, and W. M. P. van der Aalst, "Supporting condentiality in process mining using abstraction and encryption," in *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, and 9th International Symposium, SIMPDA 2019, Revised Selected Papers*, 2019.

[68] M. Rafiei, L. von Waldthausen, and W. M. van der Aalst, "Ensuring confidentiality in process mining.," *SIMPDA*, vol. 18, pp. 3–17, 2018.

[69] G. Tillem, Z. Erkin, and R. L. Lagendijk, "Mining encrypted software logs using alpha algorithm," in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017* (P. Samarati, M. S. Obaidat, and E. Cabello, eds.), pp. 267–274, SciTePress, 2017.

[70] M. Rafiei and W. M. van der Aalst, "Privacy-preserving data publishing in process mining," in *International Conference on Business Process Management*, pp. 122–138, Springer, 2020.

[71] M. Rafiei and W. M. P. van der Aalst, "Practical aspect of privacy-preserving data publishing in process mining," in *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2020 co-located with the 18th International Conference on Business Process Management (BPM 2020)*, CEUR-WS.org, 2020.

[72] A. Pika, M. T. Wynn, S. Budiono, A. H. Ter Hofstede, W. M. van der Aalst, and H. A. Reijers, "Privacy-preserving process mining in healthcare," *International journal of environmental research and public health*, vol. 17, no. 5, p. 1612, 2020.

[73] M. Rafiei and W. M. P. van der Aalst, "Towards quantifying privacy in process mining," in *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers* (S. J. J. Leemans and H. Leopold, eds.), vol. 406 of *Lecture Notes in Business Information Processing*, pp. 385–397, Springer, 2020.

[74] K. Maatouk and F. Mannhardt, "Quantifying the re-identification risk in published process models," in *International Conference on Process Mining*, pp. 382–394, Springer, 2022.

[75] M. Kabierski, S. A. Fahrenkrog-Petersen, and M. Weidlich, "Privacy-aware process performance indicators: Framework and release mechanisms," in *Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings* (M. L. Rosa, S. W. Sadiq, and E. Teniente, eds.), vol. 12751 of *Lecture Notes in Computer Science*, pp. 19–36, Springer, 2021.

[76] F. Rösel, S. A. Fahrenkrog-Petersen, H. van der Aa, and M. Weidlich, "A distance measure for privacy-preserving process mining based on feature learning," in *Business Process Management Workshops - BPM 2021 International Workshops, Rome, Italy, September 6-10, 2021, Revised Selected Papers* (A. Marrella and B. Weber, eds.), vol. 436 of *Lecture Notes in Business Information Processing*, pp. 73–85, Springer, 2021.

[77] M. Rafiei and W. M. P. van der Aalst, "Privacy-preserving continuous event data publishing," in *Business Process Management Forum - BPM Forum 2021, Rome, Italy, September 06-10, 2021, Proceedings* (A. Polyvyanyy, M. T. Wynn, A. V. Looy, and M. Reichert, eds.), vol. 427 of *Lecture Notes in Business Information Processing*, pp. 178–194, Springer, 2021.

[78] R. Zaman and M. Hassani, "On enabling GDPR compliance in business processes through data-driven solutions," *SN Comput. Sci.*, vol. 1, no. 4, p. 210, 2020.

[79] R. Zaman, M. Hassani, and B. F. van Dongen, "Data minimisation as privacy and trust instrument in business processes," in *Business Process*

Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers (A. del-Río-Ortega, H. Leopold, and F. M. Santoro, eds.), vol. 397 of *Lecture Notes in Business Information Processing*, pp. 17–29, Springer, 2020.

[80] M. Rafiei and W. M. P. van der Aalst, "Mining roles from event logs while preserving privacy," in *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*, pp. 676–689, Springer, 2019.

[81] D. Reißner, R. Conforti, M. Dumas, M. L. Rosa, and A. Armas-Cervantes, "Scalable conformance checking of business processes," in *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I* (H. Panetto, C. Debruyne, W. Gaaloul, M. P. Papazoglou, A. Paschke, C. A. Ardagna, and R. Meersman, eds.), vol. 10573 of *Lecture Notes in Computer Science*, pp. 607–627, Springer, 2017.

[82] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy, accuracy, and consistency too: a holistic solution to contingency table release," in *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China* (L. Libkin, ed.), pp. 273–282, ACM, 2007.

[83] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proc. of ACM SIGMOD*, pp. 193–204, 2011.

[84] P. Laud, A. Pankova, and M. Pettai, "A framework of metrics for differential privacy from local sensitivity," *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 2, pp. 175–208, 2020.

[85] A. Pankova and P. Laud, "Interpreting epsilon of differential privacy in terms of advantage in guessing or approximating sensitive attributes," in *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*, pp. 96–111, IEEE Computer Society, 2022.

[86] G. Elkoumy, A. Pankova, and M. Dumas, "Mine Me but Don't Single Me Out: Supplementary Material," Mar. 2021.

[87] G. Elkoumy, A. Pankova, and M. Dumas, "Differentially private release of event logs for process mining," *CoRR*, vol. abs/2201.03010, 2022.

[88] Z. Jorgensen, T. Yu, and G. Cormode, "Conservative or liberal? personalized differential privacy," in *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015* (J. Gehrke, W. Lehner, K. Shim, S. K. Cha, and G. M. Lohman, eds.), pp. 1023–1034, IEEE Computer Society, 2015.

[89] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.

[90] V. Leno, A. Augusto, M. Dumas, M. La Rosa, F. M. Maggi, and A. Polyvyanyy, "Discovering executable routine specifications from user interaction logs," *CoRR*, vol. abs/2106.13446, 2021.

[91] A. Ramdas, N. G. Trillos, and M. Cuturi, "On wasserstein two-sample testing and related families of nonparametric tests," *Entropy*, vol. 19, no. 2, p. 47, 2017.

[92] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 223–228, ACM, 2004.

[93] B. van Dongen, "BPI Challenge 2012," 2012.

[94] W. Steemanvan, "BPI challenge 2013," 2014.

[95] B. B. van Dongen, "BPI challenge 2014," 2014.

[96] B. B. van Dongen, "BPI challenge 2015," 2015.

[97] B. van Dongen, "BPI Challenge 2017," 2017.

[98] B. van Dongen and F. F. Borchert, "BPI Challenge 2018," 2018.

[99] B. van Dongen, "BPI Challenge 2019," 2019.

[100] B. B. van Dongen, "BPI challenge 2020," 2020.

[101] J. Munoz-Gama, R. R. de la Fuente, M. M. Sepúlveda, and R. R. Fuentes, "Conformance Checking Challenge 2019 (CCC19)," 2019.

[102] A. Djedović, "Credit Requirement Event Logs," 2017.

[103] F. Mannhardt, "Hospital Billing - Event Log," 2017.

[104] F. Mannhardt, "Sepsis cases-event log," *Eindhoven University of Technology. Dataset*, pp. 227–228, 2016.

[105] M. M. de Leoni and F. Mannhardt, "Road Traffic Fine Management Process," 2015.

[106] P. Gunst, "Urineweginfectie (UWI-casus) logboek," 2020.

[107] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 6280–6290, 2018.

[108] C. Dwork, G. N. Rothblum, and S. P. Vadhan, "Boosting and differential privacy," in *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pp. 51–60, IEEE Computer Society, 2010.

[109] Y. Zhu and Y. Wang, "Poission subsampled rényi differential privacy," in *ICML*, vol. 97 of *Proceedings of Machine Learning Research*, pp. 7634–7642, PMLR, 2019.

[110] K. Chaudhuri and N. Mishra, "When random sampling preserves privacy," in *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings* (C. Dwork, ed.), vol. 4117 of *Lecture Notes in Computer Science*, pp. 198–213, Springer, 2006.

[111] Y. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled renyi differential privacy and analytical moments accountant," in *AISTATS*, vol. 89 of *Proceedings of Machine Learning Research*, pp. 1226–1235, PMLR, 2019.

[112] M. Bauer, A. Senderovich, A. Gal, L. Grunske, and M. Weidlich, "How much event data is enough? A statistical framework for process discovery," in *Advanced Information Systems Engineering - 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings* (J. Krogstie and H. A. Reijers, eds.), vol. 10816 of *Lecture Notes in Computer Science*, pp. 239–256, Springer, 2018.

[113] I. Mironov, "Rényi differential privacy," in *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pp. 263–275, IEEE Computer Society, 2017.

[114] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi, "Practically efficient multi-party sorting protocols from comparison sort algorithms," in *Information Security and Cryptology - ICISC 2012 - 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers*, pp. 202–216, 2012.

[115] P. Laud and A. Pankova, "Privacy-preserving frequent itemset mining for sparse and dense data," in *Nordic Conference on Secure IT Systems*, pp. 139–155, Springer, 2017.

[116] L. Kamm, *Privacy-preserving statistical analysis using secure multi-party computation*. PhD thesis, University of Tartu, 2015.

[117] M. Borkowski, W. Fdhila, M. Nardelli, S. Rinderle-Ma, and S. Schulte, "Event-based failure prediction in distributed business processes," *Information Systems*, 2017.

ACKNOWLEDGEMENT

I would like to thank my supervisor Marlon Dumas for all the time he assigned to me and for guiding me during my PhD journey. Also, I would like to thank Alisa Pankova for her constant support and feedback.

I also would like to thank my parents, Nafisa Marie and Mahmoud Elkoumy, for all of their continuous unconditional support, especially for all the prayers I got from my mother.

I would like to acknowledge the European Research Council (PIX project) and the ERDF via the Estonian Centre of Excellence in Computer Science (EXCITE) for funding my studies. All the experiments presented in this thesis have been executed at the High-Performance Computing Center at the University of Tartu. Also, I would like to thank the city of Tartu for being such a wonderful place that boosts my productivity.

SISUKOKKUVÕTE

Privaatsuskaitse tehnoloogiaid äriprotsesside kaeveks

Protsessikaeve tehnikad võimaldavad organisatsioonidel analüüsida äriprotsesside täitmise logijälgi eesmärgiga tuvastada võimalusi oma operatiivse jõudluse parendamiseks. Protsessikaeve tehnikate rakendatavus sõltub äriprotsesside täitmise andmeid sisaldavate sündmuslogide kättesaadavusest. Mõningatel juhtudel, eriti klientidele suunatud protsesside puhul, võivad need sündmuslogid sisaldada privaatset informatsiooni ja sellest tulenevalt kohalduvad analüüsile andmekaitsemäärustest tulenevad piirangud. Sellistel juhtudel peavad organisatsioonid rakendama privaatsuskaitse tehnoloogiaid (PET), et leida tasakaal sündmuslogi analüüsimisest saadava kasu ja andmekaitsemäärustest tulenevate nõuete vahel. Eelkõige tuleb minimiseerida isikute desanonüümimise riski andmete protsessianalüütikule teatavaks tegemisel. PET annab siinkohal organisatsioonidele võimaluse sündmuslogide analüüsiks, ilma et analüütik saaks sündmuslogist üksikisikuid eraldi tuvastada. Näiteks võimaldab PET tervishoiu sündmuslogi põhjal leida protsessi kitsaskohti ja parendusvõimalusi, samas tuvastamata konkreetseid patsiente, kelle jaoks protsessi täidetakse. Antud kontekstis vastab käesolev lõputöö küsimusele kuidas anonüümida sündmuslogi ja teostada protsessikaevet selliselt, et üksikisikud, kelle jaoks protsessi täidetakse, ei oleks tuvastatavad. Olemasolevas teaduskirjanduses on erinevaid privaatsust säilitavaid protsessikaeve lahendusi välja pakutud. Üheks levinumaks on k-anonüümsusele tuginevad tehnikad. Need tehnikad eemaldavad andmestikust terveid juhtumeid või üksikuid sündmusi (tegevuste instantsid) ja sellest tulenevalt kahjustavad tulemuseks saadud anonüümitud sündmuslogi kasutatavust. Lisaks ei ole k-anonüümsusele tuginevad tehnikad turvalised predikaatide põhjal üksikisikute eraldi tuvastamise rünnete vastu. Siinkohal oluline PET on diferentsiaalprivaatsus, mis võimaldab ennetada üksikisikute eraldi tuvastamise ründeid ning pakub koosluskindlaid privaatsusgarantiisid. Diferentsiaalprivaatsus töötab lisades andmetesse müra, mille ulatus määratakse privaatsuse eelarve parameetriga ε. Hetkel parimad diferentsiaalprivaatsusele tuginevad sündmuslogi anonüümimise lähenemised kahjustavad sündmuslogi kasutatavust lisades sündmuslogisse logijälgi, mida reaalsuses ei esine. Lisaks ei paku need lähenemised juhiseid sobiva ε väärtuse hindamiseks. Käesolev lõputöö lahendab antud uurimustühimiku pakkudes välja lähenemise, mis garanteerib et üksikisiku ründaja poolt tuvastamise tõenäosus anonüümitud sündmuslogi väljastamisel ei ületa ettenähtud läviväärtust, ning et erinevus tegeliku ja anonüümitud sündmuslogi ajaliste näitajate vahel on minimaalne. Pakutud ja olemasolevate parimate lähenemiste empiiriline võrdlus, kasutades 14 reaalseid andmeid sisaldavat sündmuslogi, näitab et pakutud lähenemine annab paremaid tulemusi nii andmete kasutatavuse kui ka arvutusliku efektiivsuse seisukohast. Tavaliselt kasutatavate diferentsiaalprivaatsuse lähenemiste hindamine näitab, et esialgsele (kõrge kasutatavusega) sündmuslogile võimalikult sarnase ja samas privaatsusgarantiisid säilitava anonüümitud sündmuslogi loomiseks on

vajalik seadistada privaatsuse eelarve parameetri ε. Hiljutised diferentsiaalprivaatsuse teemalised teadustööd on näidanud, et paremat privaatsuse ja kasutatavuse tasakaalu on võimalik saavutada rakendades osavalimite võtmist enne müra sündmuslogisse lisamist. Ehk lihtsustatult, osavalimite abil on võimalik privaatsust suurendada. Käesolev lõputöö pakub sündmuslogi anonüümimise lähenemise, mis tugineb sellele tähelepanekule. Pakutud lähenemine võtab sündmuslogis sisalduvatest logijälgedest mitu valimit, lisab igasse valimisse eraldiseisvalt müra (samas säilitades valimis statistiliselt olulised logijäljed) ning seejärel komponeerib antud valimid diferentsiaalprivaatseks sündmuslogiks. Antud lõputöö sisaldab lisaks ka empiirilist kvaliteedi hindamist, mis näitab et pakutud lähenemine viib oluliselt kõrgema kasutatavuseni samade privaatsusgarantiide juures, võrreldes olemasolevate lähenemistega. Peavoolu protsessikaeve tööriistad on disainitud organisatsioonisiseste olude jaoks, lähtudes eeldusest et sündmuslogi on töötlemiseks tervikuna kättesaadav. Selliste tööriistade kasutamine organisatsioonide vahelistes oludes on raskendatud kuna organisatsioonide vahelised protsessid hõlmavad iseseisvaid osapooli kes ei soovi (või kellel ei ole juriidiliselt lubatud) detailseid sündmuslogisid üksteisega jagada. Sellest tulenevalt pakub käesolev lõputöö lähenemise ühise protsessikaeve artefakti loomiseks ja pärimiseks. Täpsemalt on tegemist sageduse ja ajalise infoga annoteeritud otsejärgnevuste graafiga, mis on loodud erinevatele osapooltele kuuluvate sündmuslogide põhjal selliselt, et osapooled ei jaga sündmuslogisid üksteisega. Pakutud lähenemine kasutab turvalise ühisarvutuse tehnoloogiat.

CURRICULUM VITAE

Personal data

Name:	Gamal Elkoumy
Date of Birth:	01.01.1991
Citizenship:	Egyptian
Language:	Arabic, and English

Education

2018–2022	Doctor of philosophy program in Computer Science – University of Tartu
2015–2018	Master's degree in Computer Engineering – Tanta University
2018–2012	Bachelor's degree in Computer Engineering – Tanta University

Employment

2018–	Junior Research Fellow – University of Tartu
2017–2018	Research Assistant – Wireless Research Center, Alexandria University
2016 –2017	Senior Research and Development Engineer, InnoTech Co., Egypt.
2012 – 2015	Junior Data Warehouse Specialist, Data Gear Co, Egypt.

Scientific work

Main fields of interest:

- Privacy-Preserving Process Mining
- Process Mining

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Gamal Elkoumy
Sünniaeg:	01.01.1991
Kodakondsus:	Egiptlane
Keelteoskus:	Araabia, Inglise

Haridus

2018–2022	Tartu Ülikooli ühine doktoriõpe informaatika erialal
2015–2018	Tanta Ülikooli, tarkvaratehnika magistriõpe
2008–2012	Tanta Ülikooli, informaatika bakalaureuseõpe

Teenistuskäik

2018–	Tartu Ülikooli, Nooremteadur
2017–2018	Alexandria Ülikooli, Research Assistant
2016 –2017	InnoTech, Teadus- ja arendusinsener.
2012 – 2015	Data Gear, Andmelao spetsialist.

Teadustegevus

Peamised uurimisvaldkonnad:

- Privaatsuse säilitamine protsessi kaevandamine
- protsessi kaevandamine

LIST OF ORIGINAL PUBLICATIONS

Publications in the scope of the thesis

1. **Elkoumy, G.**, Fahrenkrog-Petersen, S. A., Dumas, M., Laud, P., Pankova, A., Weidlich, M. (2020). Secure multi-party computation for inter-organizational process mining. In *Enterprise, Business-Process and Information Systems Modeling* (pp. 166-181). Springer, Cham.
   Lead author. The author performed the implementation and the analysis of the experiments and contributed substantially to the ideas and the writing.
2. **Elkoumy, G.**, Fahrenkrog-Petersen, S. A., Dumas, M., Laud, P., Pankova, A., Weidlich, M. (2020). Shareprom: A Tool for Privacy-Preserving Inter-Organizational Process Mining. In *International Conference on Business Process Management (BPM) 2020 (PhD/Demos)*, 2673, 72-76.
   Lead author. The author performed the implementation and the analysis of the experiments and contributed substantially to the ideas and the writing.
3. **Elkoumy, G.**, Pankova, A., Dumas, M. (2021, March). Mine me but don't single me out: Differentially private event logs for process mining. In *3rd International Conference on Process Mining (ICPM) 2021* (pp. 80-87). IEEE.
   Lead author. The author performed the implementation and the analysis of the experiments and contributed substantially to the ideas and the writing.
4. **Elkoumy, G.**, Dumas, M. (2022). Libra: High-Utility Anonymization of Event Logs for Process Mining via Subsampling. In *4th International Conference on Process Mining (ICPM) 2022*. IEEE. (In press).
   Lead author. The author performed the implementation and the analysis of the experiments and contributed substantially to the ideas and the writing.
5. **Elkoumy, G.**, Pankova, A., Dumas, M. (2022). Amun: A tool for Differentially Private Release of Event Logs for Process Mining In *International Conference on Process Mining (ICPM) 2022 (Demos)* (In press).
   Lead author. The author performed the implementation and the analysis of the experiments and contributed substantially to the ideas and the writing.
6. **Elkoumy, G.**, Fahrenkrog-Petersen, S. A., Sani, M. F., Koschmider, A., Mannhardt, F., Von Voigt, S. N., Waldthausen, L. V. (2021). Privacy and confidentiality in process mining: threats and research challenges. In *ACM Transactions on Management Information System (TMIS)*, 13(1), 1-17.
   All the authors contributed equally to this research.

Publications out of the scope of the thesis

1. Rafiei, M., **Elkoumy, G.**, van der Aalst, W. M. (2022). Quantifying Temporal Privacy Leakage in Continuous Event Data Publishing. In International

Conference on Cooperative Information Systems (pp. 75-94). Springer, Cham. *The author contributed substantially to the ideas and the writing*.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.
26. **Tauno Palts.** A Model for Assessing Computational Thinking Skills. Tartu 2021, 134 p.
27. **Liis Kolberg.** Developing and applying bioinformatics tools for gene expression data interpretation. Tartu 2021, 195 p.
28. **Dmytro Fishman.** Developing a data analysis pipeline for automated protein profiling in immunology. Tartu 2021, 155 p.
29. **Ivo Kubjas.** Algebraic Approaches to Problems Arising in Decentralized Systems. Tartu 2021, 120 p.
30. **Hina Anwar.** Towards Greener Software Engineering Using Software Analytics. Tartu 2021, 186 p.
31. **Veronika Plotnikova.** FIN-DM: A Data Mining Process for the Financial Services. Tartu 2021, 197 p.
32. **Manuel Camargo.** Automated Discovery of Business Process Simulation Models From Event Logs: A Hybrid Process Mining and Deep Learning Approach. Tartu 2021, 130 p.
33. **Volodymyr Leno.** Robotic Process Mining: Accelerating the Adoption of Robotic Process Automation. Tartu 2021, 119 p.
34. **Kristjan Krips.** Privacy and Coercion-Resistance in Voting. Tartu 2022, 173 p.
35. **Elizaveta Yankovskaya.** Quality Estimation through Attention. Tartu 2022, 115 p.
36. **Mubashar Iqbal.** Reference Framework for Managing Security Risks Using Blockchain. Tartu 2022, 203 p.
37. **Jakob Mass.** Process Management for Internet of Mobile Things. Tartu 2022, 151 p.