Anonymization Techniques for Privacy-preserving Process Mining

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humoldt-Universität zu Berlin

von
Stephan Fahrenkrog-Petersen, M.Sc.

Präsidentin der Humoldt-Universität zu Berlin:
Prof. Dr. Julia von Blumenthal

Dekanin der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Caren Tischendorf

Gutachter/innen:	1.	Prof. Dr. Matthias Weidlich
	2.	prof. dr. Wil van der Aalst
	3.	Prof. Minseok Song, PhD
Tag der mündlichen Prüfung:		30.06.2023

© Stephan A. Fahrenkrog-Petersen
all rights reserved, 2023

Für meine Eltern. Für die gesamte Unterstützung bis hierher.

Acknowledgments

First, I need to thank Matthias Weidlich. In the last 5 years he was a great mentor. This thesis would not be possible without his support and amazing supervision. His constant supported helped me to develop my research skills. I can really say that I learned a lot over my PhD.

Next, I want to thank Han van der Aa and Martin Kabierski. With both of them, I worked very closely on the ideas covered within this thesis. I was a pleasure to work with both of you!

Also, I want to thank Marlon Dumas and Arik Senderovich who hosted me on research visits abroad. In really enjoyed the time I spend in Estonia and Canada. It allowed me to get new perspectives on research and the world overall.

In my time as a PhD student, I had the chance to work with great collaborators and colleagues! I want to thank all of them for the great ride. A big thanks goes out to Agnes Koschmider, Ali Kaan Tutak, Alexandra Tichauer, Alisa Pankova, Bo Zhao, Chris Beck, Dominik Janssen, Fabian Rösel, Fabrizio Maggi, Felix Mannhardt, Felix Oesinghaus, Florian Tschorch, Gamal Elkoumy, Glenn Dittmann, Henrik Kirchmann, Hermann Stolte, Irene Teinemaa, Leopold von Waldthausen, Maike Basmer, Majid Rafiei, Massimiliano de Leoni, Mohammadreza Fani Sani, Niek Tax, Nour Elfaramawy, Peeter Laud, Rebecca Sattler, Samira Akili, Saskia Nuñez von Voigt, Steven Purtzel. I probably forgot a few, so to all of you and all the people I forgot: Thank you! I really enjoyed all our discussions related to research or other topics.

Last but not least. The journey would have not been possible without all my friends and family. Thank you, for your support no matter when, where, and how.

Zusammenfassung

Process Mining ist ein Teilgebiet des Data Minings, welches sich mit der datengetriebenen Analyse von Geschäftsprozessen befasst. Während der Ausführung des Geschäftsprozesses werden Events erfasst, wobei jedes Event für die Ausführung einer Aktivität steht. Eine Sequenz von Events, auch als Trace bezeichnet, erfasst eine Prozessinstanz. Ein Event Log bündelt mehrere Traces zu einem Datensatz. Traces entsprechen oft einer bestimmten Person, wie z.B. einem Patienten, der in einer Notaufnahme behandelt wird. Oft enthalten Traces sensible Informationen über diese Personen. Folglich ist es notwendig den Datenschutz der Beteiligten zu berücksichtigen.

Als Teil dieser Arbeit werden potentielle Datenschutzbedrohungen für Event Logs und Process Mining im Rahmen einer qualitativen Analyse diskutiert. Außerdem wird das Risiko der Re-Identifizierung von Personen anhand öffentlicher Event Logs untersucht. Dabei zeigt sich ein hohes Risiko für eine Re-Identifizierung. Um diese und andere Bedrohungen für den Datenschutz anzugehen eignet sich Anonymisierung. Die Anonymisierung von Event Logs ist herausfordernd, weil bestimmte Verhaltensmuster für die Prozessanalyse erhalten werden müssen. Dies führt zu einem sogenannten Privacy-Utility-Trade-Off, der durch Algorithmen für die Anonymisierung beachtet werden muss.

In Rahmen dieser Arbeit werden neue Algorithmen für die Anonymisierung eingeführt, die etablierte Datenschutzgarantien wie k-Anonymity und Differential Privacy für Event Logs gewährleisten und gleichzeitig bessere Utility als existierende Techniken bieten. Im Kontext von Differential Privacy werden zwei Algorithmen für die Anonymisierung des Prozessverhaltens vorgestellt: SaCoFa und SaPa. Beide Techniken anonymisieren durch das Hinzufügen von Noise, wobei angestrebt wird die Semantik des ursprünglichen Prozessverhalten zu erhalten. Außerdem wird mit PRIPEL ein Ansatz für die Anonymisierung von Event-Kontextinformationen vorgestellt. Dadurch wird die Veröffentlichung von vollständig mit Differential Privacy geschützten Event Logs ermöglicht. Auch Algorithmen, welche k-Anonymity garantieren, werden diskutiert. Konkret wird die Familie der PRETSA-Algorithmen eingeführt. Diese Familie anonymisiert Event Logs mit der Absicht, dass sie syntaktisch ähnlich zu den originalen Daten sind. Die Grundlage für diese Anonymisierung ist die Repräsentation des Event Logs als Präfix-Baum, in dem alle Traces, die gegen die Datenschutzgarantie verstoßen, mit den ihnen ähnlichsten Traces vereinigt werden. Alle neuen Algorithmen werden anhand von realen Event Logs evaluiert. Dabei zeigt sich, dass sie im Vergleich zu bisherigen Techniken bessere Utility-Erhaltung erzielen.

Abstract

Process mining is an emerging subfield of data mining that focuses on the data-driven analysis of business processes. It uses event data recorded while executing the business process, where each execution of an activity is captured by an event. A sequence of events, referred to as a trace, captures the behavior of a single process instance. An event log forms the dataset that bundles the traces belonging to one process. Traces often correspond to a specific individual, such as a patient that is treated in an emergency room. Therefore, traces can contain sensitive information about individuals such as patients, customers, or process workers. Due to ethical and legal considerations it is necessary to address these privacy issues.

In this thesis, we discuss the privacy threats that stem from trace data and the process mining setting in general through a qualitative analysis. Furthermore, we provide an estimate of the risk of re-identifying an individual by conducting a study on publicly available event logs. We show that the re-identification risk of event data is high, but re-identification is only one of several privacy threats. To address these issues, we discuss how anonymization can be used to protect sensitive information. Anonymization of event logs is challenging, as behavioral characteristics need to be preserved for process analysis. That leads to a privacy-utility-trade-off, which needs to be addressed by anonymization techniques.

In this thesis, we present novel algorithms that ensure established privacy guarantees such as k-anonymity and differential privacy for event logs while also providing utility improvements over the current state of the art. Namely, we introduce two approaches, SaCoFa and SaPa, for the anonymization of traces based on noise insertion for differential privacy. These techniques are targeted towards the control flow, the sequence of process activities that is covered by the trace. They aim to preserve the semantics of the original control flow for the anonymized data. We also provide an approach called PRIPEL that enriches these anonymized control flows with the remaining contextual information of the traces. This allows to publish a complete anonymized event log that is protected by differential privacy. In terms of anonymization for k-anonymity, we introduce a family of algorithms, the PRETSA algorithms, that ensure anonymized logs that are syntactically close to the original data. The basis for this anonymization is a prefix representation of the event log, where traces that are violating the privacy guarantee are merged with similar traces. We evaluate all our algorithms on real-world event logs and show how they provide improved utility compared to state-of-the-art algorithms.

Contents

Listing of figures

Listing of tables

Part I:

Foundations

1 Introduction

Process Mining [149] is concerned with the data-driven analysis of business processes. From a high-level perspective, process mining sits at the interconnection of data science and process science. The field relies on data that is recorded in information systems while the business processes are executed. Process mining techniques can be used for tasks such as the generation of models of the business process [150], finding conformance violations [20], or predicting future properties of an ongoing process execution [87]. However, it was shown in various studies, that process mining can also be applied to other behavioral phenomena, such as commutes [178], smart home usage [144], and network logs [98]. The data used for process mining is saved in event logs. It captures each execution of a business process in a trace, consisting of the events that record each activity that is part of the business process.

Processes are widespread in organizations [123] and are for example used to hire new employees, treat patients, and distribute financial payouts of welfare programs. As such, they relate to situations that, are closely liked to private decisions and actions. Therefore, the aforementioned event logs can capture sensible information of patients, customers, and welfare recipients. In recent years, data protection gained more and more attention in the regulatory space, leading to new privacy regulations, such as the EU's GDPR, South Korea's Personal Information Protection Act, or Brazil's Lei Geral de Proteçao de Dados. The novel regulations led to new requirements for data management within organizations. One prominent example is the right to be forgotten, that is part of the GDPR [180]. This means, that individuals can insist that their private data needs to be deleted from datasets and models, i.e. in machine learning, that are based on their data.

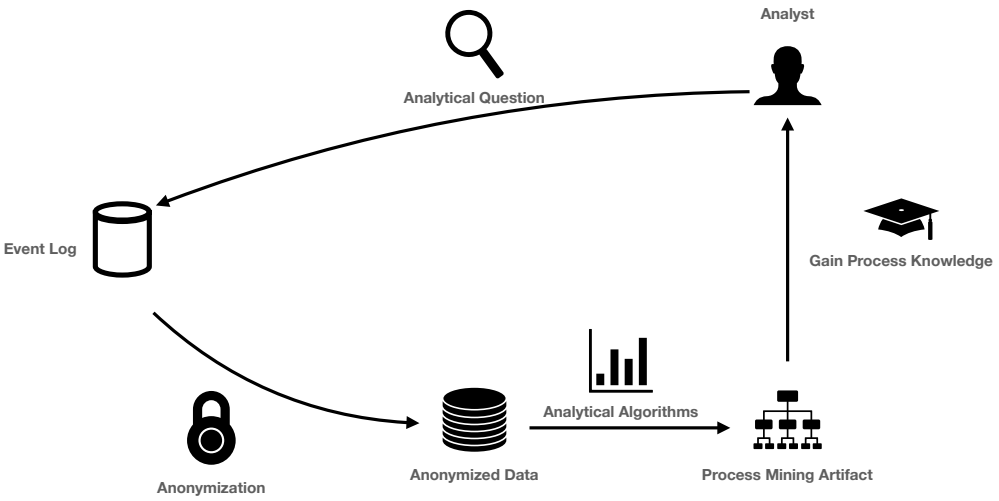


Figure 1.1: Privacy-preserving Process Mining Overview.

Consequently, privacy protection becomes of greater importance for organizations, when using data outside of the original scope of its collection. So called *secondary use* describes the use of personal data for another purpose than for which it was originally collected [93]. As an example, imagine a hospital that collects data while treating someone in an Emergency Department. The respective data was collected for the primary purpose of performing the required medical treatment, but not to analyze the respective process [140]. Furthermore, since the process data also contains information about work tasks executed by the service providers [39], such as nurses or doctors, their privacy rights need to be consider, when applying process mining to event logs. As a result the application of process mining can lead to an increasing amount of compliance needs to prevent the privacy violations, that can lead to tremendous financial fines. In case of the GDPR such fines can reach up to 4% of the annual revenue of an organization [174].

However, anonymization can be useful to manage certain privacy issues [61], as shown in Fig. 1.1, because anonymized data falls out of the scope of the GDPR [166]. Therefore, the adoption of anonymization techniques can enable process mining projects, which are otherwise considered to be too risky, due to privacy issues. This observation leads to the question of how to anonymize event logs. Traditional approaches for anonymizing tabular data are not easily applicable, because of the sequential nature of event logs, enriched with temporal and contextual information. Moreover, it is important to adjust an anonymization

technique towards the analytical purpose of the data. The main reason being that privacy-protections typically leads to a loss in utility of the anonymized data. Therefore, an analyst faces struggles with a privacy-utility-trade off [15].

Within this thesis we address this research gap. We contribute towards the greater understanding of anonymization within process mining by providing a detailed analysis of how anonymization can help to achieve privacy requirements. Furthermore, we provide several anonymization techniques that enhance the state of the art in several dimensions, i.e. by providing better utility.

1.1 Contributions

This thesis focuses on anonymization techniques for process mining data,with the aim of answering a vast set of process analytics questions. Such research questions can be asked with respect to the service consumer of a business process, i.e. how can we decrease the waiting times of a customer, or with respect to the providers of the business process, i.e. what roles exist for service providers based on the activities they perform. Within this thesis, we provide anonymization techniques to protect both of these groups of process stakeholders. Furthermore, we contribute towards a deeper understanding of anonymization of event logs in general, e.g. by providing qualitative discussions about our algorithms and naming potential angles to address their limitations. Finally, the case for the need of anonymization is made, both through, a quantitative and qualitative analysis of privacy risks. We can drill-down our contributions as follows:

1. We provide a qualitative analysis of privacy threats and the resulting requirements in the context of process mining. We also show how anonymization can be used to address these requirements. Additionally, we conduct a quantitative analysis of the re-identification risk posed in public event logs

2. We provide two algorithms, *SaCoFa* and *SaPa*, that enable control-flow anonymization that incorporates process semantics and ensures differential privacy. These techniques, focus on providing privacy for a service consumer represented by a specific case, i.e., a patient.

3. Through *PRIPEL*, we provide a framework that enable the release of differentially private event logs that include not only control-flow information, but also contextual information. Based on techniques for control-flow anonymization, PRIPEL enriches

Table 1.1: Overview of Design Space for Anonymization for Event Logs.

Privacy Target	**Control-flow Anonymization**	
	Preserve Variants	Obfuscated Variants
Intra-case (Service Consumer)	Algorithms that protect contextual information, i.e. by oversampling as it was discussed in Elkoumy et al.[45].	Algorithms that change the control-flow of a case, through noise insertion. An example for such an algorithm is SaCoFa (see Chapter 6).
Inter-case (Service Provider)	Algorithms that merge or generalize traces based on distance metrics to achieve k-anonymity, i.e. PRETSA(see Chapter 8)	Algorithms that substitute or generalize control-flows such as discussed by Rafiei et al. [121].

the anonymized control-flow with additional data from the original event log. Therefore, through PRIPEL is becomes possible to publish a complete event log, instead of just anonymized control-flow data.

4. With the *PRETSA* algorithms family, we enable the anonymization of event logs while protecting the privacy of service providers through k-anonymity and t-closeness. At the same time these techniques generate event logs that are syntactically close to the original data. We also take a deeper look into the privacy guarantees given to service providers that stretch over several cases.

The algorithms introduced in this thesis are part of a broad design space (see Table 1.1) of anonymization techniques for process mining. This design space is mainly determined by two dimensions. The first dimension is the privacy target that defines the individuals that shall be protected, either service consumers (i.e. patients or customers) or by the service providers (i.e. doctors or customer care agents). This decision depends on the question, where the anonymized data is about to be published. If the data is published to the general public or to staff that usually would be unauthorized to access the users data, the main concern is about the service consumers. However, if the data is mainly used for internal purposes, the main concern might be with the workers within the organization. Usually, a service consumer is defined by an single case for example a patient visiting an emergency room, so that protection for them focused on guaranteeing privacy for a single case. On the other hand, it in most scenarios service providers(i.e. doctors) handle multiple cases (i.e. patients). Therefore, it is necessary to consider an inter-case perspective to protect them against privacy leakage. It is important to note, that these assumptions can change, depending on how the case is defined

while creating the event log, i.e. if the workday of a service provider is used as the basis for one case.

The second dimension to consider is concerned with the preservation of the process control-flow. Depending on both privacy and utility considerations, it might be desirable to either to ensure that the anonymized data only contains variants that have been part of the original event log or to allow for the insertion of new, noisy trace variants. Having only variants from the original log provides benefits in terms of utility, since it gives certainty that all observed behavior was an actual process execution. This allows for analysis such as the checking for conformance violations such as checking if an emergency department followed clinical guidelines. However, this property also comes with downsides. If we assume that the control-flow can contain sensitive information, such as the treatment for a specific disease like HIV, an adversary might be able to learn this sensitive information from the anonymized data. This can especially be the case, if the adversary has knowledge about an unique sub-sequence of a case. Such background knowledge might enable the adversary to link a trace or variant in the published data to a specific individual.

Besides these explicitly mentioned dimensions, other considerations may inform the selection of an anonymization technique. A prominent example is the given privacy guarantee that can be driven by regulation or user-requirements. We address this issue by discussing techniques for both, differential privacy and k-anonymity, two of the most well-known privacy guarantees. Other considerations can be guided by the analysis that is to been performed on the anonymized data. This is necessary, since all anonymization leads to utility-loss and different techniques can nudge the loss in a direction, that preserves the utility for specific process mining tasks.

In addition to the conceptual contributions, this thesis also provides several artifacts. All algorithms discussed within this thesis have been made publicly available through GitHub under the MIT license, an open source license that allows for a broad usage of the respective algorithms. Furthermore, the algorithms SaCoFa and PRIPEL have been integrated into PM4Py [71], one of the leading process mining frameworks.

To achieve the contributions of this thesis, we relied on the design science method [167], an established method in the research field of process mining. The main idea behind this method is that research should lead to new artifacts. These artifacts shall be produced to solve a relevant problem, such as to enable the release of anonymized control-flows information with high utility. Furthermore, the proposed artifacts need to be evaluated and communicated, as we did through publishing at least one paper about each contribution of this thesis. Both

the method for this thesis and the problem were previously proposed and peer-reviewed as a doctoral consortium paper [49].

Notably, part of research covered in this thesis was already included in one of the standard textbooks in the field of process mining. Namely, the algorithms PRETSA, SaCoFa and PRIPEL, have been discussed in the section on *Responsible Process Mining* [90] of the *Process Mining Handbook* [156].

1.2 Statement regarding Involvement of Co-Authors

Within this section, we outline the contributions of all authors that have been involved in the papers that are the backbone of this thesis.

The work presented in Chapter 4 is based on a previously published paper [44]. All authors contributed equally towards conceptualization and writing of the manuscript. The role of this chapter is to motive the anonymization techniques that are the key contribution of this thesis.

In Chapter 5, we present work that is based on a previously published paper [168]. Saskia Nunez von Voigt participated in the conceptualization of the approach and performed most of the implementation work. She also conducted the experiments for the paper and was involved in the writing process. Stephan Fahrenkrog-Petersen was involved in the conceptualization of the approach and in the writing of the paper. Dominik Janssen also contributed to the implementation and writing of the paper. Agnes Koschmider, Florian Tschorsch, and Matthias Weidlich supervised the research and participated in the writing process. Olaf Landsiedel and Felix Mannhardt provided feedback and assisted with the writing. This chapter is an additional motivational chapter for the anonymization techniques.

Chapter 6 discusses research that is based on a previously published conference paper [50] and its journal extension [51]. The algorithms were conceptualized and implemented by Stephan Fahrenkrog-Petersen. He also performed the experiments and wrote the first draft of the manuscript. Martin Kabierski participated in the conceptualization and implementation of a preliminary version of the algorithms, and helped with writing the paper. The writing of the paper was supported by Han van der Aa. He also provided feedback regarding the algorithms. Matthias Weidlich supervised the work and supported the writing of the manuscript. Fabian Rösel was a co-author of the conference version of the paper. He supported the implementation of the SaCoFa algorithm. The bachelor thesis of Felix Oesinghaus inspired the SaPa algorithm.

The paper [55] formed the basis for Chapter 7. Stephan Fahrenkrog-Petersen conceptualized and implemented the algorithm. Also the experiments were performed by him and he drafted the first version of the paper. Han van der Aa provided feedback on the concept and was involved in writing the paper. The research was supervised by Matthias Weidlich. He also supported the writing of the paper.

The work covered in Chapter 8 builds on a previously published paper [54] and its journal extension [53]. Stephan Fahrenkrog-Petersen conceptualized and implemented the algorithms in this paper, as well performed the experiments and created the first draft of the paper. Han van der Aa contributed both with feedback on the concept and by being involved in the writing of the paper. Matthias Weidlich contributed with supervision and by writing.

1.3 Outline

- In Chapter 2, we outline basic concepts of privacy, such as the privacy guarantees k-anonymity and ε-differential privacy. This chapter also gives a high-level overview of the field of process mining.
- In Chapter 3, we give an overview of related work. We put the respective papers in perspective with our field and provide an overview about the field of anonymization for event logs in general.
- In Chapter 4, we introduce an overview of the threats and challenges regarding privacy in process mining. Additionally, this chapter outlines requirements that exist for privacy-preserving process mining. Furthermore, we explain what role anonymization can play when addressing these threats and challenges.
- In Chapter 5, we provide a novel quantification method for the re-identification risk posed by event logs. We empirically study this privacy threat using public event logs.
- In Chapter 6, we propose two control-flow anonymization techniques, *SaCoFa* and *SaPa*. They provide a differential privacy-based protection for event logs.
- In Chapter 7, we present *PRIPEL*. It enriches anonymized control-flow data with contextual data, thereby allowing for a more detailed analysis of the underlying process. The contextual data is protected by local differential privacy.

- In Chapter 8, we introduce our approach *PRETSA*. It anonymized event logs while guaranteeing k-anonymity and t-closeness. At the same time, it ensures that all anonymized behavior was already present in the original log. Additionally, it supports the anonymization of temporal information assigned to events.
- Finally, in Chapter 9, we summarize the contributions of this thesis. We also outline the limitations of our approaches and point towards directions for future research.

Each of the chapters 4 to 8 is based on at least one research publication. At the beginning of each chapter we will mention the relevant publications. At the end of this thesis, we outline the contribution from the author of this thesis to each of these publications.

2 Background

Within this chapter. we outline basic concepts related to this thesis. First, we give an introduction of the basics of process mining in Section 2.1. Followed by an outline of basic privacy notions, such as *k*-anonymity and differential privacy, in Section 2.2.

2.1 Process Mining

Organizations are run through the execution of business processes. The purpose of these business processes is to organize the work performed within an organization, aimed at supporting the strategic goals of this organization, e.g. treating patients or produces cars. The discipline studying these processes is called *Business Process Management* (BPM) [34]. The main aim of BPM is to optimize the business processes of an organization, i.e. to reduce the time necessary to perform a process or to reduce operational costs [11].

Nowadays, these business processes are executed through the usage of *Information Systems*. These systems allow *process workers* to participate in the business process, e.g. a doctor can order laboratory test or record a diagnosis. Information systems come in many forms, most prominently, as *Enterprise Resource Planing* (ERP) or *Customer Relations Management* (CRM) systems. These systems can for example support a sales process. In a ERP system a company could keep a record about the available products and place an order to refill its stock, when necessary. While a CRM system can track records about a customers buying history, allowing the salesperson to find cross-selling opportunities. However, also domain

specific information systems exist. For example, in a hospital [95] the following types of information systems could be used, a *Hospital Information Systems* (HIS), *Laboratory Information System* (LIS) or *Picture Archiving and Communication System* (PACS). Within a HIS all clinical information related to a patient can be stored and used by different clinicians within different departments. On the other hand a LIS supports the laboratory for example be providing it with the information about requested tests for a specific blood sample. Similarly, a PACS is used by the radiology department to manage medical imaging.

However, all of these information systems have in common, that they store the information related to business processes, usually in relational database systems [150]. This data forms the basis for process mining [149]. A group of techniques, at the the intersection of process science and data science. Essentially, it uses data mining techniques, to analyze *business processes*. Therefore, it can be utilized to retrieve insights about the business process. Nonetheless, first it is necessary to transform the data from the Information systems into a so called event log [152], a process mining dataset. The remaining section will provide a further outline of these topics. Within Section 2.1.1 we will provide formal notations and a example for an event logs. In Section 2.1.2, we will go into more details about process mining algorithms.

2.1.1 Event Log

Event logs captures the information about the execution of a business process, an example related to an emergency department [140] is shown in Table 2.1. Each single execution is captured as a *case*, consisting of *events*, each representing the execution of one *activity*. In the remainder of the thesis we will use the following formal model, to describe event logs:

Our event model builds upon a set of *activities* $\mathcal{A}$. An *event* recorded by an information system, denoted by e, is assumed to be related to the execution of one of these activities, which is written as $e.a \in \mathcal{A}$. By $\mathcal{E}$, we denote the universe of all events. Further, we define a function to assign the attributes to an event, the so called *event attributes* $d_e : \mathcal{E} \mapsto 2^{\mathcal{D}}$, with $\mathcal{D}$ being the universe of all possible attributes. With the notation $e_1.d_1$ we denote the attribute value of d_1 for the event e_1. Such an attribute could for example for an event of the activity *Antibiotics* be the attribute *Drug*, that reflects the prescribed medication, see Table 2.1. Usually, each event e comes with an attribute called *timestamp*, denoted by $e.ts$, that models the time of execution of the respective activity according to some totally ordered time domain.

A single execution of a process, i.e., a case, is represented by a *trace*. This is a sequence $\xi = \langle e_1, \ldots, e_n \rangle$ of events $e_i \in \mathcal{E}, 1 \leq i \leq n$, such that no event occurs in more than one

Table 2.1: An Example of an Event log.

Case ID	Activity	Timestamp	Case Attributes	Event Attributes
1000	Registration	03/03/21 23:40	{Age: 26, Sex: m}	{Arrival: check-in, Provider: Nurse A}
1000	Triage	03/04/21 00:27	{Age: 26, Sex: m}	{Status: Uncritical, Provider: DoctorX}
1000	Liquid	03/04/21 00:47	{Age: 26, Sex: m}	{ Liquid: NaCl 0.5l, Provider: Nurse A}
...	...	...	...	...
1001	Registration	03/04/21 00:01	{Age: 78, Sex: f}	{Arrival: Ambulance, Provider: Nurse B}
1001	Triage	03/04/21 00:05	{Age: 78, Sex: f}	{Status: Critical, Provider: DoctorY}
1001	Antibiotics	03/04/21 00:09	{Age: 78, Sex: f}	{Drug: Penicillin, Provider: DoctorY}
...	...	...	...	...

trace and the events are ordered by their position within the trace, usually derived from their timestamp. We adopt a standard notation for sequences, i.e., $\xi(i) = e_i$ for the i-th element and $|\xi| = n$ for the length. The universe of all traces is denoted as $\mathcal{T}$. A trace can come with *case attributes*, that relate to a whole trace and not just a single event, such as the *Age* of a patient in our example in Table 2.1. These case attributes can are denoted similar to the event attributes with $d_\xi : \mathcal{T} \mapsto 2^{\mathcal{D}}$ being a function to assign the set of case attributes. Moreover, $\xi_1.d_1$ is the notation for the case attribute d_1 of the trace ξ_1. An *event log* is a set of traces, $L = \{\xi_1, \ldots, \xi_n\}$, and we write $\mathcal{L}$ for the universe of event logs.

2.1.2 Process Mining Algorithms

The previously shown model of event logs already give information about the business process, more insights can be generated by applying process mining techniques.. In general, algorithms in process mining are categorized into three broad areas, namely *discovery*, *conformance checking*, and *enhancement* [157]. Each of them can be differentiated by its aim and different analytical results. In Fig. 2.1, we show an overview about the different techniques, which are summarized as follows:

Discovery describes the generation of a process model from an event log. Widely spread is the process of generating process models [5] that describe the control-flow of a business process. Giving information about the ordering of activities and also highlighting other relations, such as concurrency or exclusivity of activities. An example for such a process model, generated by Apromore [73], is shown in Fig. 2.2. The shown model is a so called *directly-follows-graph*. One common, but limited presentation of a business process [154]. It represents how often an activity is followed by another activity. More expressive models are petri nets [124], as represented in Fig. 2.3.

Conformance Checking uses a normative model and checks if the real-world executions of the business process as captured in the event log, violate this model [20]. A typical use case would be identifying cases within the event log that violate behavioral rules of the model [171].

Enhancement requires a model and an event log, The event log is used to adjust the model. A use case of enhancement is the automatic repair of process models [48]. Another common use case is the simulation of the business process based on a discovered model [19].

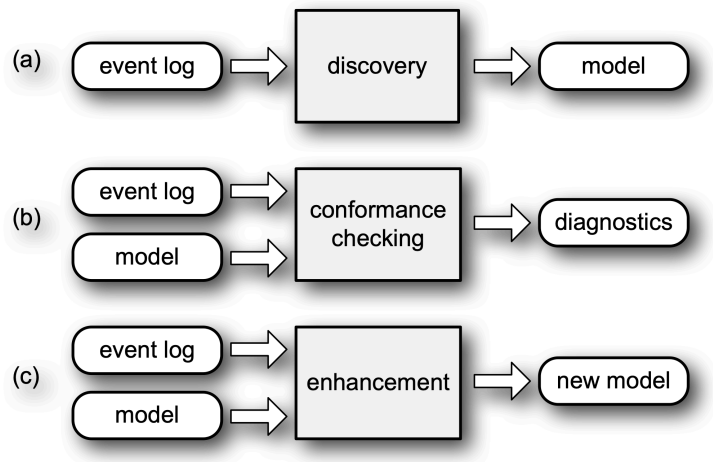


Figure 2.1: Types of Process Mining [158].

2.2 Privacy

The core idea behind privacy is the right of an individual to select which information should be shared with whom. It builds on the assumption that data about oneself should be owned

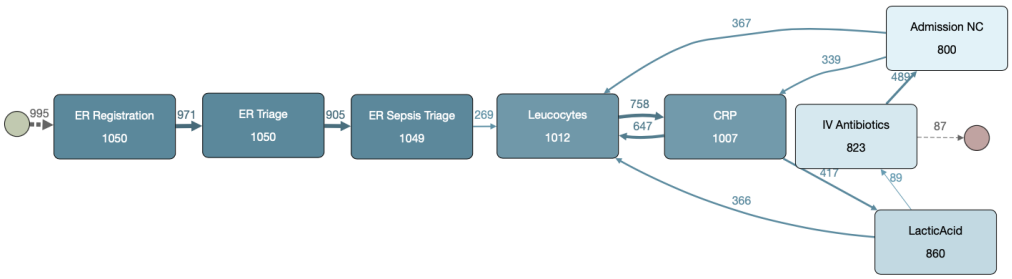


Figure 2.2: Example of a Directly-follows-Graph as a Process Model.

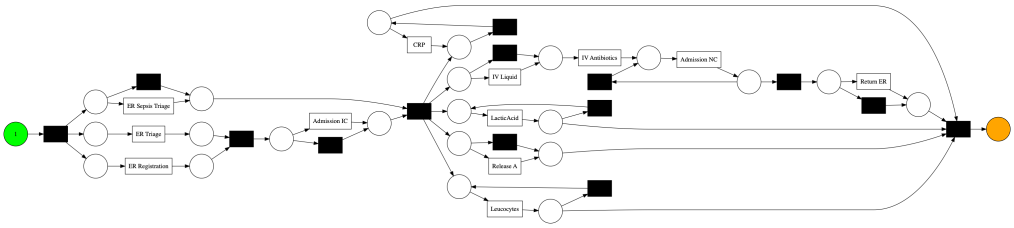


Figure 2.3: Example of a Petri Net as a Process Model.

by oneself. With regards to data analysis, this lead to the expectation that the data of individuals should be protected while being processed. A widely adopted solution to ensure this protection is the application of technical privacy guarantees [169]. These notions bound the information that can be learned about one individual from the data.

Several approaches to construct such guarantees have been proposed. Within this thesis we put emphasis on two of the most common concepts. On the one hand, we consider concepts, based on the idea of data similarity, which lead to privacy guarantees that group individuals together, so that it is no longer possible to differentiate between individuals. These guarantees are known as *k-anonymity* [141] and its extensions. In Section 2.2.1 we give details about these guarantees.

Furthermore, we study the idea of *differential privacy* [35], which is based on the idea of minimizing the effect one individual has on a dataset. Therefore, limiting the privacy loss that would occur if this individual becomes part of the dataset. The foundations of this guarantee are outlined in Section 2.2.2.

However, while these guarantees give a descriptive requirement to ensure privacy, they do not describe how to achieve them self. Therefore, we also outline a range of strategies for *data anonymization*. To give an overview about the advantages and disadvantages of the different

approaches for data anonymization. We describe the respective ideas in Section 2.2.2.

2.2.1 K-ANONYMITY

One fundamental idea to ensure privacy is to make one individual hide within a group. Consequently, it is harder to gain information of the individual. This simple idea is realized by k-anonymity. The goal of k-anonymity is to ensure that one individual is part of a group with at least k member, that can not be distinguished from each other.

To enable the formation of these groups k-anonymity splits the attributes associated with an individual into three groups: (i) *identifiers* that identify a individual directly, like a name or patient id; (ii) *quasi-identifiers*, if several of these are combined they can identify an individual like age, sex or date of birth; (iii) *sensitive attributes* that should be hidden from an adversary such as a diagnosis. We call individuals with the same quasi-identifiers an *equivalence class*.

As mentioned above, k-anonymity requires that at least k individuals are within each equivalence class. Consequently, if a dataset does not fulfill k-anonymity from the start, it is necessary to transform it in such a way, that this property is fulfilled.

Traditionally, k-anonymity was defined for database tables. In Table 2.2, we show a table and a corresponding k-anonymized table with $k = 2$, meaning that in this table each individual can not be differentiated from at least one other individual. Due to the protection several types of attacks can be prevented. Namely, *identity disclosure* and *membership disclosure*. The first means that an adversary is not able to identify which entry in a k-anonymized dataset belongs to one specific individual, since several entries have the same quasi-identifiers. *Membership disclosure* protects the dataset from knowing, if an individual's data is included in the dataset. Since a dataset is transformed in such a way, that equivalence classes are formed, e.g. by abstracting from specific zip codes, it is often not possible to determine if one specific individuals data is included in the dataset.

However, our example shows that a dataset protected by k-anonymity can still reveal information about an individual. In our example, we know that both men have HIV. If we would know the age, sex and postcode of a patient within this equivalence class, we could be sure that the patient has HIV. This privacy issue is called *attribute disclosure*, since it reveals attribute data about individuals. To tackle this issue, several extensions of the notion of k-anonymity have been proposed. Most prominently, l-diversity [86] and t-closeness [80] offer additional protection for *attribute disclosure*. In the case of l-diversity this is ensured, by enforcing an additional minimal number of values l to the sensitives attributes. For example enforcing with $l = 2$ that each equivalence class contains HIV positive and negative patients.

Table 2.2: Example of a k-anonymized Table.

Age	Sex	Zip-code	Disease
20-30	M	1*	HIV
20-30	M	1*	HIV
50-60	F	2*	Breast Cancer
50-60	F	2*	Pneumococcus

However, even such a protection can have problems, since if only one patient in a large equivalence class is HIV negative, it is still reasonable to assume that the individual under attack has HIV. Therefore, *t*-closeness limits the difference in the distribution of each equivalence class and the overall distribution. As a result the attacker can only achieve a limited information gain.

Another important aspect of *k*-anonymity is that it gets harder to achieve as more attributes can be considered as quasi-identifiers. Since each attribute potentially, splits up the individuals into smaller equivalence classes. This phenomenon can lead to high information loss for anonymized datasets resulting in the so called *curse of dimensionality* [2], making it difficult to apply *k*-anonymity to high dimensional datasets.

2.2.2 Differential Privacy

Differential Privacy [35] is based on the idea that the influence of one individual on one query should be limited. This goal is commonly achieved by introducing noise to the respective query. The character of the noise is determined by a mechanism, that should aim to optimize for high utility of the data.

Similar to *k*-anonymity differential privacy can be parameterized. On the one hand, ε provides a parameter to adjust the strength of the privacy guarantee, with lower values for ε representing stronger privacy guarantees. On the other hand, the strength also depends on the sensitivity Δf, representing the maximal impact one individual can have on the result of a query q. In a query q, counting the patients within a certain zip-code the sensitivity would for example be $\Delta f = 1$, because each patient is counted at most once. Differential privacy aims at minimizing the impact of one individual on the final result, therefore it focuses on *neighboring datasets*, meaning that two datasets ds_1 and ds_2 differ only by one entry. If we now assume that $\mathcal{DS}$ is the universe of all datasets and $\mathcal{Q}$ is the universe of all query outputs, we can formalize differential privacy as follows: For all neighboring datasets $ds_1, ds_2 \in DS$

and all subsets $\rho \subseteq \mathcal{Q}$ it holds that:

$$\Pr[q(ds_1) \in \rho] \leq \exp(\varepsilon) \cdot \Pr[q(ds_2) \in \rho] \tag{2.1}$$

One important principle of differential privacy is the idea of *parallel composition* [136]. The basic idea is that if disjoint data that is protected by ε-differential privacy is joint with other ε-differential private data the data overall is still protected by ε-differential privacy. Therefore, it is possible to access the different parts of a dataset through differential private queries and combine the results without incurring any additional privacy loss, as long as the data is disjoint.

Taking this principle even further is the idea of *local differential privacy* [175]. Its basic principle is that the data of each individual is anonymized independently, instead of the whole dataset. Therefore, allowing the applications of an infinite number of queries on the anonymized dataset. Since contrary to parallel composition the anonymization of the same data multiple times with ε-differential privacy infers an additional privacy loss. This process is called *sequential composition* [183] and weakens the data protection to $(n * \varepsilon)$-differential privacy with n being the number of times the data was anonymized.

2.2.3 Anonymization

As mentioned above different notions of privacy protection exist. At the same time different strategies to achieve anonymization exist [29]. Common strategies include *Suppression*, *Generalization*, and *Noise Insertion*. The easiest to understand is suppression it is based on the idea to remove data that could be used to identify an individual. However, it results in considerable loss of utility, since part of the data is lost for the analysis.

On the contrary, *generalization* [77] is a common way to ensure privacy notions such as k-anonymity. Through this anonymization approach attributes are generalized by using an abstraction hierarchy. For example, instead of the birthday of a patient the respective data could be generalized to the birth month, birth year or even the decade of birth. However, it is important to notice that generalization requires hierarchies that might not be available for all kind of data. While such hierarchies are usually obvious for numeric values or locations, building hierarchies for activities in a business process is a complex task [165]. So far event abstraction was mainly studied to ensure that activities are on the same level of abstraction [7].

The idea behind *noise insertion* is to change parts of the original data, until it fulfills a privacy notion. This strategy is commonly used for differential privacy. It does not require hier-

archies and also does not lead to the overall loss of certain parts of the data. However, it might lead to false conclusion since the noise can influence the analysis performed on the noisy data. We will detail out two common noise insertion mechanisms for differential privacy, because they play a major role in algorithms presented later within this thesis

Laplace mechanism

The *Laplace mechanism* inserts noise based on a Laplacian distribution and is a common mechanism for differential privacy. The impact of this mechanism generally depends on the strength of the privacy guarantee ε and the *sensitivity* Δf of some query q. A query $\hat{q}$ protected by the Laplace mechanism can formally be described as:

$$\hat{q} \leftarrow q + Lap(\frac{\Delta f}{\varepsilon}) \tag{2.2}$$

The sensitivity Δf depends on the maximum impact one individual can have on the result of query q. So, if q is a counting query as introduced above and one individual participates in at most once, the sensitivity is $\Delta f = 1$. If an individual can appear multiple times, the sensitivity is higher and more noise needs to be introduced to achieve ε-differential privacy. However, in such scenarios, the guarantee of ε-differential privacy may also be relaxed, which lowers the increase in sensitivity and still provides a relatively strong protection [67]. Furthermore, it is important that the Laplace mechanism is only one particular approach to achieve differential privacy and other approaches exist.

Exponential mechanism

The exponential mechanism [96] enables prioritization of certain query results by incorporating the notion of a score function into the noise-insertion process. This score function $s(d, r)$ defines some results to be more desirable than others for the given dataset over which the query is evaluated. Put differently, the higher $s(d, r)$, the more desirable query result r is for the dataset d. Moreover, let Δs be the sensitivity of score function s, i.e., the maximum differences between scores assigned to the possible results for any two neighboring datasets. Then, for some query q over dataset d and privacy parameter ε, the query $\hat{q}$ protected by the exponential mechanism is derived by selecting result r with a probability proportional to $e^{(\varepsilon s(d,r))/(2\Delta s)}$.

3
Related Work

In this chapter, we review related work on privacy-aware data processing., To this end, we start with work on privacy-preserving process mining in Section 3.1. Other related disciplines and how they relate to privacy-preserving process mining are discussed in Section 3.2. In Section 3.3, we finish this chapter with a conclusion that discusses how this thesis fits into the related scientific work.

3.1 Privacy-related Research within Process Mining

To facilitate a comprehensive understanding, we have structured the existing literature on privacy-preserving process mining into several sub-areas. In Section 3.1.1, we review studies that discuss general issues of privacy-preserving process mining. In Section 3.1.2, we examine techniques that utilize encryption to protect data. Section 3.1.3 covers anonymization techniques that provide formal privacy guarantees, such as k-anonymity and differential privacy. Additionally, we provide an overview of techniques suited for inter-organizational process mining in Section 3.1.4. Finally, in Section 3.1.5, we discuss academic tools for privacy-preserving process mining.

3.1.1 High-level Considerations on Privacy in Process Mining

The issue of privacy within the process mining discipline was discussed in the process mining manifesto [158], published already in 2012. However, a paper by Mannhardt et al. [93] re-

established the importance of the topic in light of the GDPR. The authors pointed out, that the data collected for business process execution was usually never intended to be used for process mining. Therefore, process mining would be categorized as secondary usage, which implies privacy challenges, such ensuring the right to be forgotten and the need to obtain the consent of individuals represented in the data. ,The issue of privacy was also discussed as one challenge towards responsible process mining [153], a sub-field of process mining that is concerned with ethical aspects of process mining projects. Besides privacy, responsible process mining is also concerned with topics such as fairness and accountability. Here, it tries to prevent injustice that stems from process mining, i.e., decisions that discriminate against certain groups. The topic of privacy-preserving process mining was further investigated in terms of its importance for process mining in healthcare settings [109]. Here, the relevance of privacy considerations originates from the sensitive nature of patient information, e.g., the encoding of treatments for chronic diseases within activities such as diabetes. The topic of how anonymization can be used to tackle privacy challenges in process mining in general (instead of discussing a single technique) was first discussed by the author of this thesis [49]. In particular, we argued for anonymizing event logs and developing privacy-aware process mining techniques. The issue on how to handle the practical aspects related to the XES-standard for publishing anonymized data was also discussed within the literature [116]. The work proposed a method to document what anonymization has been applied to the data. Another consideration that has received attention is the continuous release of event log data [112, 118]. It is important to know that a continuous publication of data can lead to a higher privacy loss.

3.1.2 Encryption-based Techniques

Previous research has examined the use of encryption as a means of preserving privacy in event logs. The basic idea behind this approach is that it makes it more difficult for an adversary to gain information from encrypted data. For example, research has explored the use of homomorphic encryption to obscure sensitive information within event logs [119, 120]. Additionally, a method was developed to apply the alpha algorithm, a basic process discovery technique, to encrypted data [145].

A known threat to encryption-based techniques is the use of frequency attacks, where an adversary attempts to break the encryption by exploiting knowledge of the frequency of certain activities. However, the risk of such attacks is acknowledged in the encryption literature and countermeasures have been proposed. One such protection is the decomposition of activities into sub-activities, a strategy that was applied for privacy-preserving role mining [114].

Table 3.1: Categorization of Privacy-preserving Techniques for Event Logs.

Privacy Target	**Control-flow Anonymization**	
	Preserve Variants	Obfuscated Variants
Intra-case (Service Consumer)	TraVas [122], Elkoumy et al. [45], Event Log Encryption [119, 120]	TLKC [117], Mannhardt et al. [92], Kabierski et al. [64, 65]
Inter-case (Service Provider)	u-PPPM [9], k-PPPM [8], Privacy-preserving Role Mining [114]	TLKC [121, 117]

However, even after this a formal privacy such as k-anonymity or differential privacy can not be guaranteed and the protection can be broken if the encryption is broken.

3.1.3 Anonymization for Process Mining

The current literature on privacy protection in process mining proposes two main strategies: One is focused on privacy through hiding individuals within groups and the other limits the impact that an individual can have on the final result through differential privacy. Additionally, anonymization techniques can also vary based on the privacy target (service consumer vs. service provider) and the question of whether variants should be preserved or obfuscated. In this subsection, we will provide an overview of anonymization techniques in process mining that have not been introduced in the research covered in this thesis. We categorize the respective techniques based on their privacy target and how the anonymize the control-flow in Table 3.1. Note, that we left out techniques that do not fit into this categorization, because they are not primarily concerned with the information of specific individuals. However, we included the techniques discussed in the previous subsection

Group-based Privacy

The idea of hiding individuals within groups was studied within several research papers. Most work in this area adopts the notion of k-anonymity and it derivatives(see Section 2.2.1). One approach from this are is the technique *u-PPPM* [9] that reduces the risk of identifying service providers, by uniforming the distribution how often the different service providers perform which activity. Another approach that targets service providers is *k-PPPM* [8], which aims at preserving the privacy of service providers through k-anonymity. To achieve this goal, it clusters service providers with a similar behavior together to form equivalence classes that

satisfy *k*-anonymity. The TLKC approach [121, 117]., in turn, is suitable for both, service consumer and service provider protection. Here, TLKC also stands for a relaxed version of the *k*-anonymity guarantee, which allows to make assumptions of different types and strength of background knowledge of an adversary. Furthermore, the work introduces an algorithms that provides the TLKC guarantee through suppressing events. Such suppression can result in trace variants that have not been part of the original event log, which, as mentioned above, may be undesirable.

Differential Privacy

The idea of differential privacy for event logs was originally introduced by Mannhardt et al. [92]. Here, the authors protected control-flow queries on event logs by inserting noise based on the Laplace mechanism. In Chapter 6, we discuss some disadvantages of this strategy and how to address them. In general, this approach was targets the protecting of the privacy of service consumers and may introduce obfuscated variants. *TraVas* [122] is an approach that only publishes variants that have been included in the original event log. It does so, by selecting only a subset of variants that can be included within its output. Another approach that only publishes original variants is the work by Elkoumy et al. [45] that oversamples traces at risk and offers an additional protection for the timestamps. The protection of the timestamps is ensure by adding noise. All approaches that for the control-flow publication can be combined with the privacy amplification strategy *Libra* [41] that utilizes sampling. While, all previously mentioned work focuses on differential privacy for the process control-flow. While the work by Kabierski et al. [64, 65] showed how to protect the computation of process performance indicators.

3.1.4 Inter-organizational Process Mining Techniques

The techniques discussed so far are suited for intra-organizational process mining. This means, they consider the setting where all the data is available within the same organization. However, a lot of processes span over several organizations, which are often not able to share the data with each other. We call these processes inter-organizational processes [151]. A privacy-preserving approach to share process-related data for process mining was proposed by Elkoumy et al. [42]. In particular, the work presents an approach that uses multi-party computation to calculate a directly-follows-graph over distributed event logs. Another approach by Liu et al. [82] was proposed to directly mine inter-organizational process models

by combining private process models of organizations with shared public models of an inter-organizational process.

3.1.5 Tools for Privacy-preserving Process Mining

Several efforts have been made to make privacy-preserving process mining techniques more accessible in the form of tools. Multiple tools provide web-based interfaces for event log anonymization, such as Amun [46], ELPaaS [10], and PC4PM [113]. These tools package different anonymization techniques and are primarily focused on intra-organizational process mining. The only tool that relates to inter-organizational process mining is Shareprom [43]. Shareprom enables the mining of distributed event logs through multi-party computation.

3.2 Related Areas of Research

There are further areas of research that are closely related to privacy-preserving process mining, namely privacy-preserving data mining and privacy research related to sequence data. In Section 3.2.1, we discuss the broad field of privacy-preserving data mining and how privacy-preserving process mining fits into it. Next in Section 3.2.2, we discuss the specific case of privacy for sequence data, a field that is closely related to privacy research for process mining.

3.2.1 Privacy-preserving Data Mining

Privacy-preserving data mining has been studied widely both in general [4, 97], and also in specific settings, such as in the domain of clinical data [29]. One consideration that is central to this research area is the privacy-utility-trade-off [81]. It describes that higher levels of privacy preservation usually lead to a stronger loss in utility. Therefore, privacy-preserving data mining research aims to find a good balance of these two aspects. As for process mining, privacy is often captured by formal privacy notions [169]. Similar to privacy-preserving process mining the research on privacy-preserving data mining is usually targets specific data analysis tasks, such as association rule mining [182] or collaborative filtering [143]. It could be argued, that process mining is that one of these task and is a therefore part of the general line of research on privacy-preserving data mining.

Certain areas of research received special attention within recent times. For example, the idea of generating synthetic data [21] as a mean to ensure privacy-preserving data mining has been explored. Here, a machine learning model, i.e. a neural network, is trained with the

original data and is used to create synthetic data as an output [1]. This synthetic data shall represent data that is similar to the original data and mirrors the original characteristics and distributions. The application of such techniques for privacy-preserving process mining was only briefly outlined in [66].

Another approach that recently gained a lot of momentum is federated learning [177, 179]. Here, the data is distributed similar to the setting of inter-organizational process mining and a machine learning or data mining technique is applied on the distributed data to generate one global model. The application of federated learning for process mining was discussed in several papers [70, 155], but, so far no specific techniques have been presented.

3.2.2 Sequence Data

Privacy research for sequence data is closely related to privacy-preserving process mining, since traces used as a starting point for many process mining techniques denote multivariate sequence data. Both fields are concerned with protecting sensitive information in data, but sequence data presents unique challenges. Anonymization of sequence data is a challenging task [24] due to its high-dimensional nature. However, event sequences [60, 110] contain both timestamps and context information, as is the case with process mining data. An example of event sequence data is clickstream data from social media.

To address the challenges of anonymizing sequence data, various techniques have been proposed in the literature. One such technique is BF-P2kA [99], an anonymization technique that provides k-anonymity and utilizes a prefix-tree-based algorithm, which inspired our work on PRETSA (see Chapter 8). Another typical strategy is to cluster similar elements within a sequence together [104]. Here, similar sequences are group within a cluster and these sequences within each cluster are generalized to conform with the group-based privacy guarantee.

In addition to anonymization, privacy-preserving visualization of event sequences has also been studied [23]. These studies primarily focus on the application of group-based privacy protection strategies, such as the use of sankey diagrams [22] to represent the flow of activities. A sankey diagram can be used to visualize the flow within sequences and can be made privacy-aware by enforcing a minimal group size for the visualized information.

Another sub-type of sequences data is trajectory data. Often, this data has a hidden structure and the data is often timestamped. Nonetheless, the kind of structure is different from process mining, since it is mostly driven by geographical features such as streets. The trajectory/location data community has also studied the problem of re-identification within their

data [28] and has shown that most individuals can be re-identified through long sequences. To address this issue, the community has begun to utilize synthetic data generated by neural networks [12]. However, also traditional approaches such as differential privacy [176] or group-based privacy guarantees [146] have been adopted.

3.3 Conclusion

Privacy-preserving process mining techniques can be categorized based the kind of privacy notion they provide. Furthermore, we established that different techniques protect different individuals, either the service consumer or the service producer. In the following chapters, we contribute to these lines of research with novel anonymization techniques. Besides anonymization techniques that give a formal privacy notion, the issues of inter-organizational process mining and encryption based techniques have been studied in the literature. Several tools for privacy-preserving process mining have been introduced.

Notably, some similarities between privacy-preserving process mining and other areas of research exist. Research on privacy-preserving data mining and privacy for sequence data have been discussed as related areas. However, these areas optimize towards different notions of utility than used in process mining. Nonetheless, these areas can provide inspiration for the development of novel anonymization techniques for event logs. For instance, the PRETSA algorithm family was inspired by the BF-P2kA technique.

Part II:

The Case for Anonymization

This chapter is based on concepts and results previously published within the ACM Transactions on Management Information Systems [44].

4

Threats and Requirements for Privacy-preserving Process Mining

The processing of event logs for process mining can fall within the scope of privacy regulations such as the GDPR [93]. In this context, it is unclear what requirements and threats need to be addressed, which keeps data handling entities in doubt and at risk. Especially, it has to be clear what steps can be taken to reduce privacy risks. To address this issue, a group of experts in the field, including the author of this thesis, collected a list of threats for privacy in the process mining context. The group co-authored the research article that is the foundation for this chapter [44]. These threats form the motivation for privacy-preserving process mining and a potential solution for the derived requirements in the form of anonymization of event logs.

The remainder of the chapter is structured as follows: First, we introduce a motivating example and lay out the different layers for attacks, in Section 4.1. In a next step, we list the possible attack on the privacy of process stakeholders in such a process mining setting in Section 4.2. A list of requirements for privacy-preserving process mining follows in Section 4.3. We provide a discussion on how anonymization can help to address these threats and requirements in Section 4.4. Finally, the chapter ends with a conclusion in Section 4.5.

4.1 Motivating Example

While there is a general understanding that privacy needs to be considered in the process mining context, it is also important to spell out the actual threats and requirements. It is best to start with a concrete example, where process mining could be applied, to understand the potential threats and requirements. As an example scenario, we look at an emergency room setting. Applications in healthcare have seen process mining adoption [105, 127] and also received attention in other areas of data science [29, 83]. Therefore, the emergency room denotes a representative and understandable example. The goal of process mining in our scenario is to decrease waiting time for patients, and improve documentation by discovering cases of non-compliance. Additionally, the hospital is also interested in benchmarking, i.e., comparing how its performance differs from other hospitals.

Concretely, the hospital wants to apply process mining to discover the clinical pathways of different patients from the moment they arrive in the emergency department until their discharge from the hospital. Each visit of a patient to the hospital forms a case, and the individual events of each case are sourced from a Hospital Information System (HIS). For the benchmark, the hospital wants to share some of this data over organizational boundaries and therefore, needs to consider the privacy right of the individuals involved in the process. In our case this includes the patients and also the service providers, like the nurses and doctors referenced within the event log. Within process mining the service providers are also individuals of concern, since an event log would allow us to learn personal information about them to. An event log could for example reveal when they take vacation or how much hours they work [108]. Also it might be possible to misuse the data to perform illegally detailed surveillance of their work.

When considering the privacy issues in process mining, we need to consider all elements of the respective process mining system, namely three layers: data, application, and presentation layer. Each of these layers comes with special challenges. To understand these better, we first need to understand what exactly we mean by these layers:

Data Layer

Process mining starts from one or several event logs, each representing the execution of several instances of a business process. These event logs form the *data layer*. In Table 4.1 we show the event log for the aforementioned example. While each trace reveals detailed information about a patient, we might also be able to retrieve sensitive information about service providers

Table 4.1: Event Log Example from a Hospital.

Case ID	Activity	Timestamp	Case Attributes	Event Attributes
1000	Registration	03/03/21 23:40	{Age: 26, Sex: m}	{Arrival: check-in, Provider: Nurse A}
1000	Triage	03/04/21 00:27	{Age: 26, Sex: m}	{Status: Uncritical, Provider: DoctorX}
1000	Liquid	03/04/21 00:47	{Age: 26, Sex: m}	{ Liquid: NaCl 0.5l, Provider: Nurse A}
...	...	...	...	...
1001	Registration	03/04/21 00:01	{Age: 78, Sex: f}	{Arrival: Ambulance, Provider: Nurse B}
1001	Triage	03/04/21 00:05	{Age: 78, Sex: f}	{Status: Critical, Provider: DoctorY}
1001	Antibiotics	03/04/21 00:09	{Age: 78, Sex: f}	{Drug: Penicillin, Provider: DoctorY}
...	...	...	...	...

by looking not just at a single case, but by considering several cases that a service provider worked on. This could allow to retrieve conclusions about service provides.

Application Layer

Algorithms that process event logs and compute representations are an important part of process mining and denote the *application layer*. To perform these applications it is often necessary to consider the fine-granular data of an event log. In some scenarios it even be necessary to integrate event logs from several organizations. When this is the case we speak of *inter-organizational process mining* [151]. In such a scenario, it might necessary that the process mining is performed by a third party. Therefore, the problem arises how such an application can be performed in a privacy-preserving manner. In our example, the hospital might want to compare its emergency room with the emergency rooms of other hospitals within the same city. Yet, the hospital would want to avoid sharing the data of their patients.

Presentation Layer

The *presentation layer* consists of generated artifacts such as the discovered process models or the rule violations detected by conformance checking, or other analytical outputs. In most

cases, these artifacts are aggregated representations of the event log, e.g., a process model with projected frequencies as shown in Fig. 4.1, representing a directly-follows graph. Regarding privacy, it is important to note that these artifacts do not directly reveal the exact underlying event logs and only provide an aggregated view on the process. In many cases, process analysts could use these refined results to generate insights about the process without the need to analyze the fine-granular event data. However, originally hidden, confidential information might be revealed through the presentation [85]. In our example process model (see Fig. 4.1), the different release types are pseudonymized, with the intention of hiding the patients outcomes. However, only one of the shown activities, namely *Release B*, never leads to re-admission of the patient. Instead, in all except one case, this release always leads to the end of the case. Consequently, we can guess that this release represents the death of a patient. Such a disclosure of information was possible by only looking at the model and therefore only relying on information from the presentation layer.

4.2 Threats for Privacy in Process Mining

In this section we present threats to privacy in process mining. We collected the threats based on, first, developing a list of concrete attacks and, then, cross-referencing this list with generic attacks from the literature.

We explicitly exclude attacks from malicious adversaries that have control over the information flow, because we consider these attacks to be part of security threats. Instead, we assume an honest-but-curious attacker who follows the protocol and has legitimate access to the data. An attacker might be a process analysts who obtains sensitive information from published artifacts. Overall, we distinguish between four categories of threats: re-identification, reconstruction, membership disclosure, cryptanalysis; each of which is described in more detail below. We also instantiate the threats in the context of our hospital scenario.

4.2.1 Re-identification Threats (T1)

Re-identification or de-anonymization threats describe the risk that the identity of an individual is disclosed to an adversary. Such a threat is based on singling-out individuals from process mining artifacts such as event logs [38]. This threat was so far studied for event logs [168] and process models [85]. However, most attention was paid to event logs that form the previously introduced data layer, since event logs contain very fine-granular data about individuals. The literature knows several possible attack strategies for re-identification threats.

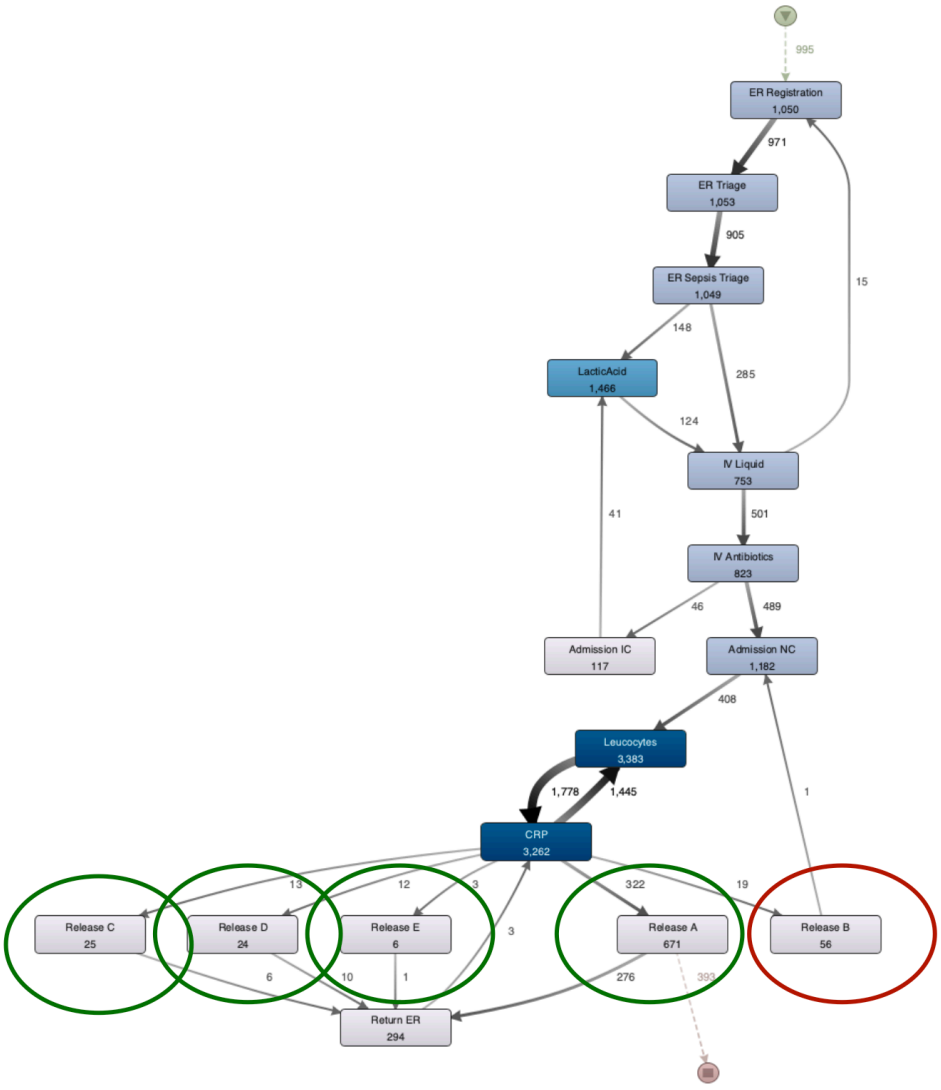


Figure 4.1: Process Model with Privacy Issue at the Activity marked with Red Circle.

A common re-identification based attack is the *linkage attack*. Here an adversary uses background knowledge and combines it with a published artifact. This, for example, could be an event log that was published after pseudonymization for research purposes. In Chapter 5, we will investigate such an attack in detail and show that serious potential for privacy leaks in published event log data.

Another commonly described attack is the *intersection attack*. An adversary can perform such an attack if several organizations independently publish event logs with overlapping populations. For example if these organizations publish data about the same inter-organizational process or when public data about individuals in the event log exists. A famous incident of such an attack led to the development of *k*-anonymity [141]. In this incident it was possible to re-identify prominent politicians within health insurance records, by combining them with public voter lists. This story shows that if an adversary knows that a target is contained in several event logs, the identity may be disclosed by taking the intersection. In our scenario, two hospitals could independently publish event logs of patient trajectories in which age is generalized to prevent a linkage attack, i.e., Table 4.1 would only contain age groups 0–20, 21–40, and so on. Patient Bob would not be easily re-identifiable anymore. However, let us assume that an adversary knows that Bob was transferred from hospital A to hospital B. If we now assume that only in one case in Bobs age group a transfer happened in the logs, it becomes possible to re-identify Bob.

4.2.2 Reconstruction Threats (T2)

The risk of (partially) reconstructing the original event log from a released process mining artifact such as a process model or a machine learning model [30] is the so-called reconstruction threat. The privacy of individuals is threatened by such an attack, because attributes belonging to certain individuals could also be reconstructed. Such attacks would mostly likely be performed against the presentation layer of a process mining application.

Let's assume an adversary failed to link one individual directly, but still aims at gaining knowledge about attributes belonging to this individual. One potential angle for an attack could be a so-called *difference attack* on aggregated statistics. Here, an adversary aims at isolating a single value through the combination of multiple aggregated statistics. For instance in our scenario the analyst can only query aggregated statistics about an event log, such as the frequencies in the process model shown in Figure 4.1. The analyst could obtain the frequency visualization grouped per release and, therefore, know the number of patients (unique cases) for each outcome. In a second query, the analyst could exclude 26-year-old patients in the

query and obtain the same statistics. From the difference, an adversary can infer the release type of Bob, the only 26-year-old patient.

Another potential attack is based on the idea of *model-inversion* [57], an adversary attempts to reconstruct data that was used to train a machine learning model. The input and outputs of the model are used to create a probabilistic version of the training data. Models used for predictive or prescriptive process monitoring [52, 87] could be vulnerable to this kind of 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 service providers treating a patient. However, so far, model-inversion attacks have not been studied in the process mining context. Therefore, it is unclear which particular risk are induced by them.

4.2.3 Membership Disclosure Threats (T3)

The threat of membership disclosure describes the risk of gaining knowledge of whether an individual is included within a certain dataset or model. In contrast or re-identification, this attack is purely about finding if an individual is part of a dataset .

An adversary might use a *membership inference attack* to determine if a certain dataset includes a specific individual. To achieve this, the adversary trains so-called shadow models to predict the membership within a released model [134]. Shadow models are used to capture the difference in misclassification between samples that are probably part of the training data and samples that are not part of the data. In a process scenario. an adversary might check if a process model allows the behavior of a certain trace and use this information to predict if the trace was included in the event log used for automatic process discovery. In our exemplary hospital setting, an adversary might use that knowledge to predict if a patient received treatment for a certain disease. This might be possible if the event log only captures certain departments of a hospital. For instance, assume an attacker knows that a target patient Alice underwent several surgeries. If the attacker has access to the process model showing that a surgery could never happen after another surgery, they can conclude that Alice is not included. Still, this knowledge may leak sensitive information. For instance, if the dataset was extracted for a set of patients with a specific disease, based on membership disclosure, we could derive that Alice does not have the respective disease.

4.2.4 Cryptanalysis Threats (T4)

A naive strategy to address privacy is pseudonymization of personal data, such as the name of the medical personal in our hospital scenario. A common alternative to pseudonymization is the encryption of the event log [120]. However, such kind of protection might be overcome by an adversary through *cryptanalysis*.

An adversary might apply a *frequency analysis attack* that exploits the characteristics of the pseudonymized data. For such an attack, background knowledge of the process can be used, e.g., knowledge about the frequency of certain activities or the position of certain activities within a trace. For example, when considering our hospital scenario, certain treatment steps, such as antibiotics, might appear more than once in a trace, while the registration and release of a patient usually happen at the beginning and the end of each trace. So assuming the activities in Table 4.1 had been encrypted, an adversary might be able to guess that the most common start activity is related to the registration of a patient. Therefore, the adversary had both the plaintext and its encrypted version. Both together could be used to break the total cipher. Here, the adversary's background knowledge may be that, for example, only the doctor *John Doe* was working on the triage on March 4, 2021. In this case, an adversary can link the activities *Triage* with the pseudonymized service providers identifier *Doctor X*. It is not difficult to gain this knowledge, especially in relatively open environments, like a hospital.

4.3 Requirements for Privacy-preserving Process Mining

In this section, we discuss requirements that address the threats discussed in Section 4.2. The requirements are taken from a systematic synthesis of the current privacy and confidentiality landscape conducted by Gharib et al. [59], who themselves based their work on a previous literature review [58]. 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:

- *R1 - Anonymity* describes the idea, that personal information can only be used in such a way that the identities of the involved individuals stay hidden [33, 107]. According to GDPR Recital 26, the protection for personal data does not need to be applied to anonymized data [106]. Therefore, guaranteeing anonymity is a big step towards compliance with privacy regulations.

- *R2 - Unlinkability* means that it should be prohibited that personal data can be linked towards the corresponding individual [107]. It complements the requirement for anonymity in the sense that preventing identity disclosure cannot be guaranteed by only making identifiers unreadable. Instead, all identifiers that could be used for linkage need to be hidden. This means that the personal data should also be sufficiently protected from re-identification through linkage attacks.

- *R3 - Unobservability* ensures that it is impossible to observe the identities of individuals that perform any action [107]. It should be noted that unlike anonymity and unlinkability, that protect the identity of the actor, the goal here is to protect the actions themselves. Therefore, the requirement is mostly relevant for the continuous monitoring and processing of data generated by running systems. Furthermore, it includes the protection of personal data against unauthorized processing, as described in Article 5 of GDPR.

- *R4 - Notice* describes that individuals are to be notified when their data is collected [33]. Such notice needs to contain a detailed description which data of the gathered data. This requirement is related to the concept of *consent* in GDPR, that regulates that data processing requires consent. According to Article 7 of the GDPR, the data processor need to be able to prove that the individuals providing their personal data have consented to its processing. Note that the consent needs to be kept updated based on the purpose of data processing.

- *R5 - Transparency* ensures that individuals should be able to know who uses their data. Furthermore, it should be possible for them to understand how their data is used and for what purposes [33]. In the GDPR the idea of transparency can be found in Recital 58.

- *R6 - Accountability* means that individuals can hold someone accountable, if they misuse information belonging to that individual [33]. This requirement covers both personal and non-personal information. Within GDPR, it can be found in Article 5, 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 should have minimal impact on the utility of the process mining techniques. Therefore, we consider three 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.
- *R8 - Application* requirements mean that the algorithms which are applied should be efficient, and fulfilling privacy 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 requirements by causing only a minima utility loss of the anonymized data and having the ability to repeat different types of queries without privacy disclosure.

4.4 The Role of Anonymization

We introduced threats and requirements for privacy-preserving process mining. However, it still remains open how these threats and requirements can be addressed. One established strategy within the privacy domain is the *anonymization* of data. As such, the question arises if a anonymization can address the threats and requirements of privacy-preserving process mining? We want to investigate this question within the following section.

First, let us look at the threats introduced above and how anonymization can mitigate them. Here, we want to emphasize anonymization with the privacy notions *differential privacy* and *k-anonymity*. The risk of re-identification (T1) is one of the main aims behind both, differential privacy and *k*-anonymity. The main goal of group-based privacy protections, such as *k*-anonymity, is to hide an individual within a group, and therefore, limit the risk of re-identification. On the other hand, differential privacy limits the impact of one individual and therefore also mitigates the risk that one individual can be re-identified. Consequently, anonymization can be an important tool to address re-identification threats. At the same time both privacy notions can offer protection from membership disclosure (T3), through *plausible deniability*, meaning that an individual can claim not to be part of a dataset. Here, the argument is that the dataset was anonymized and data supposedly matched to them, could also just be a result of the anonymization process. Furthermore, anonymization helps addressing reconstruction attacks (T2), since it limits the information an adversary can obtain from the anonymized dataset, which limits the possibility for these attacks. For example, by hiding individuals within groups. Also, the threat of cryptanalysis (T4) is important for

pseudonymized or encrypted logs, but applying mathematical rigorous privacy notions, such as differential privacy or *k*-anonymity, prevents this threat. For example, differential privacy puts a hard limit on what an adversary can learn from a dataset, independent of the background knowledge an adversary possess.

While we now understand how anonymization helps mitigating privacy threats, we also need to understand if it supports the requirements of privacy-preserving process mining. Obviously, anonymization can address the requirements of Anonymity (R1) and Unlinkability (R2), since these are the basic requirements for successful data anonymization techniques. Also, anonymization should ensure a meaningful utility of the anonymized data, and therefore, fulfills the presentation requirement (R9). However, it is impossible to address certain requirements by anonymization alone. Nonetheless, it is important to understand, that according to some privacy regulations, such as GDPR, anonymized data does not need to comply with all the requirements. Just as an example, let us consider the notice requirement (R4). While an individual needs to be notified that their data is used for data processing, such as anonymization itself, once the data of an individual was truly anonymized, it no longer falls within the scope of the GDPR [61, 106]. Therefore, a notice for all further processing of the anonymized data is unnecessary. As a consequence, it is obvious that anonymization can be helpful in also addressing most of the remaining requirements.

Furthermore, it is important to notice, that anonymizing the event log has an additional benefit. Once an event log is anonymized the privacy notions also offers protection for the application and presentation layer [49]. In contrast someone could use an privacy-preserving process mining algorithm that process the data in a privacy preserving manner and provides a privacy-protected result [115]. Such a strategy might offer better utility, but the privacy notion protects only the specific results of the algorithm, for example the generated process model. Furthermore, if several artifacts are derived for a process, combining them can lower the privacy protection, as previously discussed in the context of sequential composition in differential privacy, see Section 2.2.2. Therefore, we conclude that the anonymization of event logs has significant benefits and should be the first choice in terms of privacy protection for process mining.

4.5 Conclusion

Privacy is an important concern for many organizations. Within this chapter we gave an overview over four main threats that need to be considered in terms of privacy in a pro-

cess mining context. Furthermore, we specified concrete requirements stemming from these threats and we linked these requirements to concrete parts of the GDPR. We also described how anonymization can be used to address these threats and requirements. Specifically, we outlined how privacy protection based on k-anonymity and differential privacy can support privacy-preserving process mining. Based on this chapter we conclude that anonymization of event logs offers a powerful tool for privacy-preserving process mining that can be used to address many challenges.

This chapter is based on concepts and results previously published in [168], in the Proceedings of the International Conference on Advanced Information Systems Engineering 2020.

5 Re-identification Risk of Event Logs

While the existence of privacy threats is widely recognized, it is unclear, how dramatic they are in terms of their impact. Nonetheless, it is important that event logs are published, either for research purposes, such as to evaluate process mining techniques, or for business scenarios, for example if a hospital wants to share its data with a pharmaceutical company.

Against this background, we argue that it is crucial to understand the re-identification risks of event logs. With this insight, we can better assess the need for anonymization imposed by event logs. As a proxy for the re-identification risk, we developed a method to measure the uniqueness of cases within an event log. Furthermore, we conducted a large-scale study with 12 publicly available event logs. Our results for these logs suggest that an adversary can potentially re-identify up to all of the cases, depending on the richness of the background knowledge. We show that an adversary needs only a few attributes of a case to successfully mount such an attack.

Within this chapter, we will first motivate the need for event log re-identification risk quantification within Section 5.1. We will introduce a technique to determine the re-identification risk in Section 5.2. In Section 5.3 we apply this technique to a wide range of public event logs to generate insights regarding the re-identification risk imposed by event logs. Finally, we close the chapter with a discussion of our results and draw conclusions in Section 5.4.

5.1 Problem Illustration

The scientific literature points to many examples that show the general risk of data re-identification is a relevant threat [28, 74, 102, 103, 126, 138]. As an example, Narayanan and Shmatikov [102] de-anonymize a data set from Netflix containing movie ratings by cross-correlating multiple data sets. They modified their approach to apply it to social networks [103]. However, the re-identification risk of event logs has not received much attention yet. As shown previously, events consist of different parts, such as the activity, a timestamp, and event attributes that capture the context. Additionally, within an event log the events are ordered and partitioned into traces. Cases could also have case attributes that can contain general information about an individual, i.e., their age. All these different aspects of event logs could be potentially targeted by re-identification attacks. Therefore, it is necessary to develop specific approaches to capture the re-identification risk of event logs. The general attack behind this kind of attack is that an adversary would use background knowledge to link an individual with a case or event in the event log, e.g., by cross-correlating publicly-available sources, with the aim of extracting sensitive information. The higher the uniqueness of a case the higher the chance for an adversary to successfully link the case to their background knowledge. Therefore, we can estimate re-identification risk of event logs through measuring their uniqueness. Depending on the type of background information, different adversary models are possible, i.e. targeted at the control-flow or the case attributes. Within the remainder of this chapter, we will introduce novel uniqueness measures for process mining data and use them to measure the re-identification risk of public event logs.

5.2 Re-identifications of Event Logs

Within this section, we introduce our approach to measure the uniqueness of event logs. In Section 5.2.1, we discuss general considerations and introduce the basic idea of our approach. In Section 5.2.2, we outline how case attributes can be used for re-identification, while in Section 5.2.3, we discuss how the control-flow of a trace can be exploited to re-identify a trace.

5.2.1 Basic Idea

The basis for our uniqueness measures are two well-known approaches for uniqueness measurements [28, 126, 138]. Rocher et al. [126] estimate the population uniqueness based on

Table 5.1: Preparation of an Event Log.

Case ID	Sex	Age	Activity	Timestamp	Arrival
1000	male	26	[Reg., Triage, Liq., . . .]	[3/3/21, 3/4/21, . . .]	[check-in, Na, . . .]
1001	female	78	[Reg., Triage, Antib., . . .]	[3/4/21, 3/4/21, . . .]	[ambulance, Na, . . .]
1002	female	38	[Reg., Triage, Antib., . . .]	[3/5/21, 3/5/21, . . .]	[ambulance, Na, . . .]
. . .	. . .	. . .	. . .	. . .	. . .

given attribute values. We employ their method to estimate the uniqueness based on case attributes. Our method to estimate the uniqueness based on traces relies on the approach presented in [28, 138], where uniqueness in mobility traces with location data is estimated. Due to the sequential nature of events that are enriched with attribute values, it is necessary to apply sequence encoding [78] to the event logs. This will result in a representation of a case as one row within a database. In Table 5.1, we provide an example of such an encoded event log from an emergency department. In the applied encoding, we represent the case attributes like *sex* or *age* as columns.

Furthermore, we encode the *activity*, *timestamp*, and event attributes as columns that contain a list of the respective attribute values. The lists contain values corresponding to the order the values appeared within the events. If an event did not had a certain event attribute value, we fill the respective element in the list with *none*.

We use the uniqueness of an event log to estimate the likelihood of a successful re-identification of a case. To do so, we investigate a number of projections. These projections are a representation of a case within in an event log that often contains only a subset of the data of that case, for example only only the control-flow traces without any timestamps, case or event attributes (later on called projection E). The main idea behind projections is that a lot of process mining techniques only use parts of the event log, for example most process discovery techniques ignore attributes and timestamps [135]. Therefore, publishing an event log only partially might be justified from an analyst point of view. At the same time, it enables us to minimize the published data, which is an important principle for minimizing privacy risks. Furthermore, projections enable us to assess the risk in different scenarios, based on the intended process mining analysis. Table 5.2 summarizes the projections for event logs and their potential usage in process mining.

5.2.2 Uniqueness based on case attributes

In case of non-protected data, an individual can be identified through unique identifiers, such as a case ID or a name of a service provider. However, it is also possible to identify an individual through so called quasi-identifiers, information that can be linked to that individual. Such quasi-identifiers include information, such as birthdays, sex, or the height of a person. It was shown in the literature that the combination of quasi-identifiers can be used to identify a vast majority of individuals [141]. In a process mining setting, event logs may contain such quasi-identifiers in form of the case attributes that may contain the age or sex of a patient. It is common to quantify the re-identification risk of a dataset by measuring the uniqueness of its quasi-identifiers [25]. We adopt this strategy to a process mining setting by measuring the uniqueness of the case attributes. Let us assume that each case is associated with only one patient and each patient is associated with only one case. Consequently, the high uniqueness of a case would lead to a high risk of re-identification for the respective patient. It is important to note that usually one quasi-identifier is not sufficient to re-identify an individual. Instead, the combination of several attributes may be used to uniquely represent an individual. This is particularly the case, when attributes are linked with other datasets or sources of information. However, using multiple attributes may lead to a successful re-identification. Let us provide an example for an event log that is potentially at risk. The event log BPIC 2020 [163] contains information about travel cost reimbursements at the Eindhoven University of Technology (TU/e). At the same time, the university publishes information related to the travel of its staff on its website[*]. Linking both datasets could lead to a successful linkage attack and single out a certain individual within the event log.

As a next step, we formalize the aforementioned idea. We define the uniqueness of an event log as the relative frequency of unique cases. Let f_D be the absolute frequency of traces with the combination of values of case attributes $D \subseteq \mathcal{D}$ in an event log. One case is considered unique, if $f_D = 1$, i.e., there is no other case with the respective case attribute values. Therefore, uniqueness for case attributes is defined as

$$Uq_{\text{case}} = \frac{\sum_{\xi \in L} I_{f_D}(\xi, L)}{|L|}, \tag{5.1}$$

where the indicator function I_{f_D} is 1, if for a trace ξ it holds that $|\{\xi' \in L \mid f_D(\xi) = f_D(\xi')\}| = 1$, and otherwise I_{f_D} is 0. In our example Table 5.1, two cases have the attribute

[*] https://research.tue.nl/en/activities/ (Last time visited on March 29, 2023)

Table 5.2: Projections of Event Logs.

Projection	Data included	Exemplary usage in Process Mining
Full Event Log	All	Multi-perspective Process Mining [89]
A	activities, timestamps	queue mining [132]
B	activities, event and case attributes	predictive process monitoring [87]
C	activities, event attributes	decision mining [129]
D	activities, case attributes	trace clustering [137]
E	activities	process discovery [150]
F	case attributes	traditional data mining [3]

value "sex: female". This leads to two possible candidates for cases, and therefore, we have $f_D = 2$, which implies that the combination is not unique. However, if we additionally take the quasi-identifier *age* into account, all cases listed in our example would be unique and therefore we would have $Uq_{\text{case}} = 1$.

While we can measure the uniqueness within an event log easily, it is important to consider that often only a sample of the full event log is published. For example, only cases that appeared within a certain time frame or in a certain location. Therefore, we need to distinguish between sample uniqueness and population uniqueness. The frequency of unique cases within a published event log is called *sample uniqueness*. The term *population uniqueness* refers to the frequency of unique cases in the complete event log under the assumption that the published event log might only be a subset of the overall event log. While we can exactly measure the sample uniqueness, we can only estimate the population uniqueness, since the overall population is unknown. We refer to population uniqueness as the number of cases that are unique within the sample and are also unique in the underlying population, i.e., whole event log the from which the given log has been extracted.

Several models exist to estimate the population uniqueness based on a sample. It is necessary to estimate the overall population, so called superpopulation, based on the sample [25]. In our setting, we adopt a method of Rocher and Hendrickx [126] to estimate the population uniqueness.[†] The authors developed a model based on Gaussian copulas to estimate the population uniqueness. For this analysis, we assume that the event log is a published sample. Copulas allow to construct an overall multi-variate distribution based on the underlying

[†]Code available at `https://github.com/computationalprivacy`.

marginal distributions that we in our case estimate based of the attribute distributions of the event log. By applying the method, we can estimate the population uniqueness of cases in terms of their case attributes.

5.2.3 Uniqueness based on traces

Re-identification through case attributes is not the only potential angle of attack for an adversary. It is also possible to re-identify a case through its trace. This is especially relevant, since a lot of the public event logs for process mining do not have any, or only very few, case attributes. For example the Sepsis event log [88] has only one case attribute, that encodes the age of a patient. This is partly due to the fact that sometimes case attributes are encoded within an event log as event attributes of the first event within its trace. Furthermore, it is possible that the case attributes have been removed before publication due to privacy concerns. Moreover, it is possible that case attributes are more common in an industry setting.

Therefore, we also introduce a uniqueness metric that is based on the trace of a case. We assume that an adversary's goal is to re-identify an individual by using partial knowledge over the trace. The adversary aims at revealing sensitive information about the individual by gaining knowledge about the overall trace. Such partial knowledge can come in different forms. An adversary might know the set of activities $\{a_1, ..., a_n\} \in \mathcal{A}$ executed within a trace, the respective multi set, or the (sub)-sequence of the activities [121]. An adversary that exploits such background knowledge might be able to link certain traces within an event log to specific individuals. This is especially the case, if we assume that an adversary knows that an individual is part of a released event log. Such an assumption prevents that an adversary links a unique trace belonging to another individual to their background knowledge.

As our example in Table 5.1 shows, even without considering the case attributes, all cases are unique: case 1000 is uniquely identifiable by its third activity *liquid*. The cases 1001 and 1002 are uniquely identified by combining the activity with the respective timestamp. Given this information from the trace an adversary may identify the patient and reveal additional information from the event log.

Against this background, we define the re-identification risk as the ratio of unique traces and we apply a uniqueness measurement from the domain of location trajectories [28, 138]. In the area of location trajectories, the equivalence to an event consists of a tuple of location and timestamp. The location is equivalent to an activity in our setting. However, in process mining, events can have additional attributes that provide contextual information.

We always assume that an adversary knows an event e_i with all its attributes. We call this

knowledge a *point p*. Given a set of m random points, denoted by M_p, we check the number of traces that include the set of points. A trace is unique, if the set of points M_p is only contained in a single trace. The uniqueness of traces given M_p is defined as

$$Uq_{\text{trace}} = \frac{\sum_{\xi \in L} I_{M_p}(\xi, L)}{|L|}, \tag{5.2}$$

where $I_{M_p}(\xi, L) = 1$, if trace ξ is unique in L, i.e., $|\{\xi' \in L \mid M_p \subseteq \bigcup_{1 \leq i \leq |\xi'|} \{\xi'(i)\}\}| = 1$, otherwise $I_{M_p}(\xi, L) = 0$.

5.3 Evaluating the Risk of Public Event Logs

To explore the re-identification risk in practice, we use publicly available event logs. We only focus on event logs that capture real-life-individuals.

Certain events logs do not contain data that belongs to humans. Instead their events are the activities that have been automatically generated by software systems or robots. As an example, the NASA Crew Exploration Vehicle event log [75] consists of events from the run of an automated unit test suite at NASA. Additionally, some of these event logs only contain a single case, which makes them unsuited for our evaluation. Nonetheless, if we could link a case to a suitable identifier, we would also able to measure the uniqueness for event logs that capture software activities. For example, of the activities encode tasks that are being performed by a person. By using an appropriate transformation, this person, one of the service providers, could serve as a case identifier.

However, we only apply our methods to estimate the uniqueness of the real-life-individuals event logs. For event logs with more than one case attribute, we calculate the uniqueness of case attributes. Table 5.3 summarizes the results of our classification and shows some basic event log statistics. Furthermore, we indicate the applied uniqueness measures.

Due to ethical considerations, we will apply our methods and discuss intermediate results in detail only for the BPI Challenge 2018 [164] and the Sepsis [88] event logs. Both event logs where originally published by authors involved in the paper [168] that is the foundation for this chapter.

For the remaining event logs, we use a pseudonym and only show brief results. Note that the pseudonymized event logs in the following sections have not the same order as in Table 5.3, but the pseudonymization is consistent across the evaluation.

Table 5.3: Classification of Event Logs.

				uniqueness
event log	#cases	#activities	case attr.	traces
ADL [142]	75	34	no	yes
BPIC 2012 [160]	13,087	24	yes	yes
BPIC 2015 [161]	1,199	398	yes	yes
BPIC 2017 [162]	31,509	26	yes	yes
BPIC 2018 [164]	43,809	14	yes	yes
CCC 2019 [101]	10,035	8	no	yes
Credit [31]	20	29	no	yes
Hospital Billings [94]	100,000	18	no	yes
RlH [159]	1,143	624	no	yes
CoSeLoG [16]	1,434	27	yes	yes
Traffic Fines [27]	150,370	11	no	yes
Sepsis [88]	1,049	16	no	yes

5.3.1 Uniqueness results based on case attributes

We first look deeper into the BPIC 2018 event log. It was originates from the German company "data experts" and is based on an application of payments process of EU's Agricultural Guarantee Fund. Each case represents a payments application by one farmer, over a period of three years. In total the log consists of 43,809 cases. The log comes with the following set of attributes *payment_actual* (PYMT), *area* (ARA), *department* (DPT), *number_parcels* (#PCL), *smallfarmer* (SF), *youngfarmer* (YF), *year* (Y) and *amount_applied* (AMT) as case attributes. Here, we notice that the combination of "youngfarmer" and "year" can be used to gain information related to the birth year of the farmer, which may enable re-identification via information that relatively easy to obtain.

With this insight, we now turn to evaluate the impact of case attributes and their combinations. It is important to note that some of the case attributes have been prepossessed by grouping the values in 100 bins [164]; namely this is true for attributes PYMT, #PCL, and AMT. We still investigate which combinations of case attribute values make cases more distinct and thus unique. Intuitively, if an adversary possess more background knowledge, it becomes more likely that an individual can become identifiable.

To investigate this assumption, we count the number of unique cases, for each combination of case attributes. In this setting we can show that our intuition holds true, since the

Table 5.4: Sample uniqueness and population uniqueness (estimated) based on case attributes (left for BPI Challenge 2018; right for all event logs).

Combination	Sample	Population
PYMT	0.409	0.161
PYMT, ARA	0.476	0.164
PYMT, DPT	0.528	0.419
PYMT, #PCL	0.698	0.594
PYMT, ARA, #PCL	0.747	0.649
PYMT, DPT, #PCL	0.788	0.718
PYMT, DPT, #PCL, ARA, SF	0.845	0.971

Event Log	Sample	Population
3.	0.011	0.005
6.	0.035	0.071
7.	0.152	0.146
8.	1.000	0.952

more case attributes are known, the more unique the cases become. As shown in Table 5.4 (left), when only considering PYMT , there are 40.9% unique cases. However, when combining PYMT with #PCL, uniqueness increases to 69.8%. If an adversary would know all case attributes, 84.5% of the cases would be unique within the event log.

It is not possible to get a complete picture of the re-identification risk by just considering the sample uniqueness. Therefore, we also measure the population uniqueness by applying the method described in Section 5.2.2. In Table 5.4 (left), we present the estimated population uniqueness as the average of five runs. When considering all case attributes, a high population uniqueness is reached, of around 97%. But even when considering only the single case attribute (PYMT), a population uniqueness of 16.1% is reached. Additionally, we also calculate the sample uniqueness and estimate the population uniqueness for all event logs with more than one case attribute, in total we run our experiments on four event logs. For our analysis, we ignore case attributes that contain activities of the event log (i.e., the first executed activity), since we assume that the exact control-flow is unknown to the adversary. For the other event logs, we mostly observed low uniqueness, with the exception of log 8 (see right side of Table 5.4). For the logs with a low uniqueness, the case attributes often cover general information about the trace that are unrelated to the individual behind the case, such as the start date of the case.

5.3.2 Uniqueness results based on traces

Before we present our overall results on uniqueness based on traces, we want to provide a detailed look at the Sepsis event log. Originally, it was obtained from an HIS of a Dutch hospital. The cases capture the clinical workflow of patients. These patient come to an emergency room and are suspected to suffer from Sepsis. The treatment outcome of patients was

studied using process mining techniques by Mannhardt et al. [91]. The event log is widely used for process mining studies and was released as a public event log [88]. At the time, when the event log was released sophisticated event log anonymization techniques for event logs have not been around. Therefore, several basic steps were taken to prevent re-identification, including:

- Pseudonymization of the working diagnosis and discharge related activities, e.g., "Release B";
- Randomization of timestamps by perturbing the start of cases; and adjusting timestamps of respective subsequent events accordingly
- Generalization of employee information by stating the department only and the age of the patient to groups of 5 years with at least 10 people.

The Sepsis event log has only one case attributes, as such, it is a good candidate for the trace-based uniqueness, but not for the case attribute-based uniqueness. Due to the presence of timestamps and event attributes, it is possible to apply the previously introduced projections.This was we can to simulate both different powers of the background knowledge of the adversary and different projections.

We apply the method introduced in Section 5.2.3 to estimate the uniqueness of traces. For this technique we rely on so-called points that the adversary knows as their background knowledge. We assume that a point consists of activities, timestamps, and departments that are currently responsible for a patient's treatment. For each case, we randomly select m points of the trace and count the number of traces within the log with identical points. In other words, we look for other traces that, for example, include the same activities executed by the same department. We use the random selection of points, because we believe that it is most realistic to assume that an adversary possesses only some random points instead of knowing the exact order of executed activities. Consequently, we assume the adversary only possesses a weak form of background knowledge and we need to assume that our technique underestimates the re-identification risk in case the adversary possesses strong forms of background knowledge. Hence, a high uniqueness in our experiment shall be considered more serious, since as a more sophisticated and optimized point selection could lead to an even higher uniqueness.

We present the results for the Sepsis event log in Fig. 5.1. The figure contains the results for different projections and differing number of points known to the adversary.[‡] As expected the higher the number of points, the higher the measured uniqueness. Furthermore, the

[‡]Code available at `https://github.com/samadeusfp/re-identification-risk`.

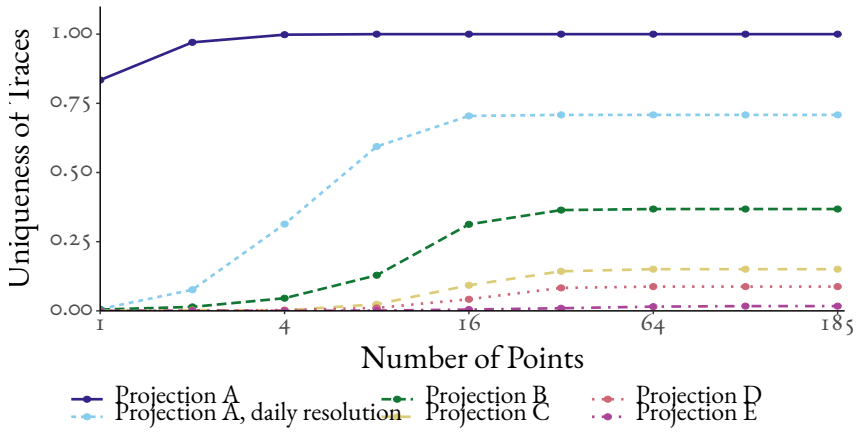


Figure 5.1: Uniqueness based on traces for Sepsis event log.

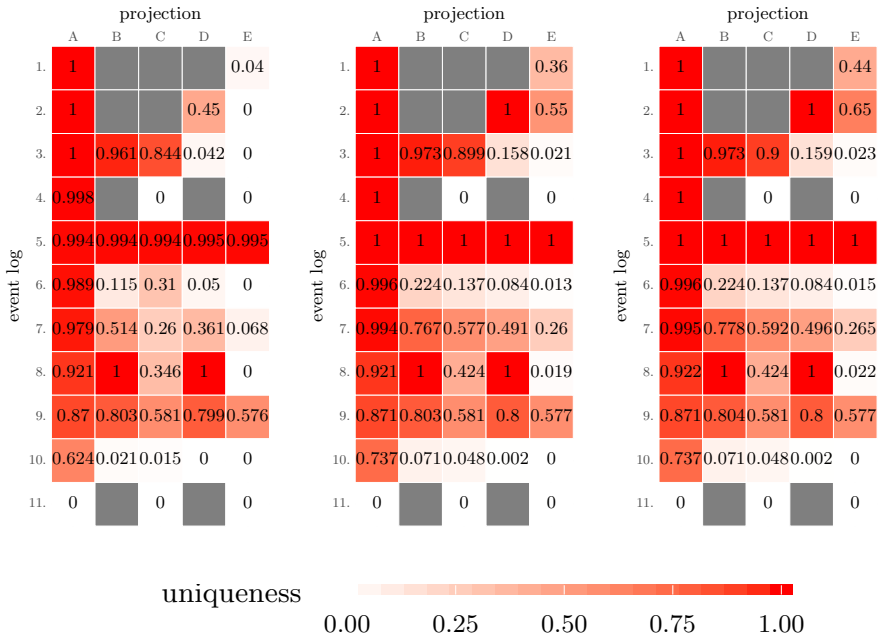
timestamps (projection A) lead to the highest re-identification risk. However, we were able to reduce the impact of timestamps by setting the resolution to days, which is included in Fig. 5.1 as a separate line.

Moreover, using projections without timestamps decreases the re-identification risks even further. This becomes clear by considering projection B that assumes the published event logs contain all activities, case and event attributes. Yet, this projection is able to significantly limit the uniqueness to approximately 37%. The uniqueness of traces remains stable for more than 64 points since only 2% of the traces have more than 64 points.

We apply our method estimating the uniqueness based on traces to all event logs categorized as event logs about real-life-individuals. We present the results for uniqueness for all the event logs and projections in Fig. 5.2. Moreover, we show how the uniqueness changes when considering different sizes of background knowledge, by providing the respective data for scenarios where 10%, 50%, and 90% of the points of each trace are known. If it was not possible to include a certain projection because of missing attributes, we highlight this with a gray cell in the figure.

For all event logs, we see a similar trend as before for the Sepsis event log, as shown in Fig. 5.2: Projection A, including timestamps, leads to the highest uniqueness. For most event logs, it is even possible to uniquely identify all or nearly all of the cases, knowing just 10% of all points. However, this high number represents the scenario, in the which the exact timestamps are known.

There is a strong reduction in the uniqueness, if the timestamps are not part of the projection. Moreover, it is a general trend that a reduction in information leads to a decrease



(a) For 10% of points. (b) For 50% of points. (c) For 90% of points.

Figure 5.2: Uniqueness based on traces for all event logs.

in uniqueness. By considering the differences between projection B and projection C, this trend becomes obvious. Here, the only difference is, that projection C does not contain case attributes, while both event logs contain the activities and event attributes. However, projection C leads to a significant reduction in terms of uniqueness compared to projection B. Only considering the activities, without further information about the attributes, leads to low uniqueness, as shown in projection E with the exception of event log 5 and 9. A potential reason for this observation is that the cases within these logs vary in terms of their control-flow.

We observe that adding points increases the uniqueness. This effect is more prevalent by increasing the number of points from 10% to 50% than from 50% to 90%. For example, the uniqueness of projection A for event log 10, when knowing 10% of the points, increases from 62.4% to 73.7%, when knowing 50% of the points. However, given 90% of points of the trace, we cannot observe an increase of the uniqueness for event log 10. The same observation can

be made for other event logs and other projections.

Overall in our empirical study, we measured higher uniqueness based on traces than on case attributes. Therefore, we conclude that knowledge of the trace, especially the timestamps, makes event logs particularly vulnerable to re-identification attacks.

5.4 Conclusion

In this chapter we introduced approaches to quantify the re-identification risk of event logs. We evaluated this risk by applying our technique to a major subset of all event logs used within the research community.

Through experiments, we could show that most event logs have a high re-identification risk. 11 out of 12 evaluated event logs showed a trace uniqueness greater than 62%, with even a limited background knowledge of only 10% of all points. We also came to the conclusion that publishing more information, e.g. timestamps or event attributes, leads to higher re-identification risk, than a more extensive background knowledge. However, our results clearly show that little background knowledge is sufficient and already induces a considerable re-identification risk for event logs. In contrast, generalization of timestamps helps to reduce the risk [184]. The results, however, show that combining several attributes, such as case attributes and activities, still lead to a significant re-identification risk. These results highlight the need for anonymization of event logs becomes obvious, since it would allow to protect the published data by giving privacy guarantees [49].

Part III:

Anonymization Techniques

This chapter is based on concepts and results previously presented in [50] the Proceedings of the International Conference on Process Mining 2021 and in [51], within Information Systems.

6

Semantics-aware Mechanisms for Control-flow Anonymization in Process Mining

This chapter addresses the problem of process control-flow anonymization, while preserving as much utility as possible. Within this chapter, we focus on providing privacy protection for the control-flows of individuals represented in a single case. The control-flow perspective denotes the basis for many process mining techniques. We capture the control-flow of all cases in an event log as the *trace-variant distribution*, which is the trace variants of an event log and their occurrence frequencies. Due to its prominent usage in process mining, the anonymization of such distribution with differential privacy was previously studied.

The state-of-the-art strategy [92] to guarantee the privacy is based on inserting noise into the trace variant distribution of a log. The resulting trace variant distribution is protected by differential privacy. This intuitively, leads to privacy protection by adding behavior, as shown Fig. 6.1. Unfortunately, the state-of-the-art approach comes with certain drawback in terms of utility: It inserts noise randomly, which neglects the semantics of the underlying process. The returned trace variants may then represent behavior that was never observed or, more importantly, which is clearly impossible for the process at hand. Additionally, the total number of traces within the anonymized trace variant distribution might differ in orders of magnitude from the original distribution.

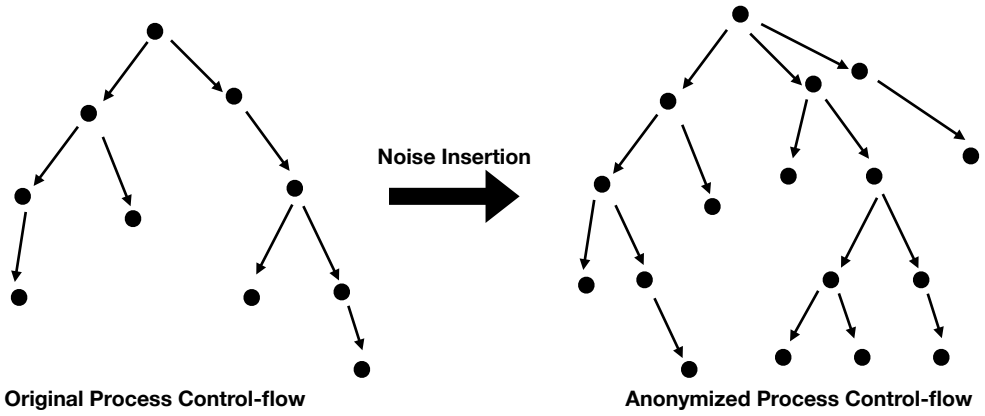


Figure 6.1: Process Control-flow visualized as a Prefix-Tree to provide an Intuition of Anonymization by Noise Insertion.

Including obviously incorrect sequences lowers the utility of the published data for process analysis, e.g., resulting in misleading models. At the same time, adversaries can easily recognize such trace variants as the result of the anonymization procedure, so that the assumed privacy guarantee no longer holds.

Against this background, we target the question of how to incorporate a process' semantics in control-flow anonymization. In this chapter, we address this question with two algorithms. First, we present SaCoFa (semantics-aware control-flow anonymization). Our idea is to achieve differential privacy of trace-variant queries based on *exponential noise-insertion* techniques. Unlike noise insertion with the Laplacian mechanism that is adopted by the state-of-the-art [92], the exponential mechanism enables us to control the way noise is inserted, while providing the same degree of privacy [36].

Second, we also present SaPa (semantics-aware play-out-based anonymization). The idea behind SaPa is to anonymize the directly-follows distribution and published a trace-variant play-out of it. Through this indirect approach, it is possible to keep the total number of traces in the anonymized distribution close to the original distribution.

Within the remainder of this chapter, we will first outline the details of the addressed problem in Section 6.1. We continue by explaining the algorithm SaCoFa in Section 6.2. Next, we turn to the details of SaPa in Section 6.3. In Section 6.4, we present an experimental evaluation of our mechanism for control-flow anonymization. We also provide a qualitative discussion of both techniques in Section 6.5. Finally, we summarize our contributions towards differential private control-flow techniques in Section 6.6.

Table 6.1: Illustration of a trace-variant distribution, both original and privatized.

(a) Original trace-variant distribution.

Trace Variant	#
⟨*Register*, *Triage*, *Surg.*, *Release*⟩	20
⟨*Register*, *Triage*, *Surg.*, *Antibio.*, *Release*⟩	12
⟨*Register*, *Triage*, *Antibio.*, *Antibio..Release*⟩	6
⟨*Register*, *Triage*, *Antibio.*, *Surg.*, *Release*⟩	5
⟨*Register*, *Triage*, *Consul.*, *Release*⟩	2
⟨*Register*, *Triage*, *Consul.*, *Surg.*, *Release*⟩	4

(b) Privatized trace-variant distribution.

Trace Variant	#
⟨*Register*, *Triage*, *Surg.*, *Release*⟩	105
⟨*Register*, *Triage*, *Antibio.*, *Antibio.*, *Release*⟩	7
⟨*Release*, *Triage*, *Triage*, *Surg.*, *Register*⟩	4

6.1 Problem Illustration

To illustrate our goal, let us revisit our example event log from an emergency room. An analyst might want to understand the clinical pathways of patients within the hospital. The intention of this analysis could be to understand common clinical pathways or to minimize the transition time of patients. For such an analysis, it is not necessary to use data except the control-flow of a business process. We therefore, neglect the information on other process perspectives, such as timestamps. We can retrieve the respective trace-variant distribution $\theta : \mathcal{A}^* \to \mathbb{N}$ through trace-variant query $\tau(L)$:

$$\tau : \mathcal{L} \to \Theta$$

with Θ being the universe of all trace-variant distributions. Aside from $\tau(L)$ as the query for all trace variants, we also define a query that returns the number of times a trace variant v occurs in L:

$$\tau : \mathcal{L} \times \mathcal{A}^* \to \mathbb{N}.$$

Alternative to a trace-variant query $\tau(L)$, an adversary could also consider the directly-follows query $\partial(L)$ of an event log. Such a query also covers information about the control-flow. It calculates how often an event e_1 corresponding to an activity $a_1 \in \mathcal{A}$ is directly-followed by an event e_2 corresponding to an activity $a_2 \in \mathcal{A}$ within the same trace ξ. We formally define a directly-follows query $\partial(L)$ as:

$$\partial : \mathcal{L} \to \Theta$$

with Θ being limited to activity sequences of a length of 2.

As shown above (see Chapter 5), the control-flow of a process can contain information that can be used to re-identify cases. In a setting like an emergency room, each trace corre-

sponds to one specific patient. Therefore, re-identifying cases can reveal sensitive information, such as the treatment for specific diseases. Obviously, such data is sensitive and should be anonymized.

The topic of anonymized trace-variant queries received attention in research. First, the issue was covered in a study by Mannhardt et al. [92]. They proposed a technique for the privatization of trace-variant queries which construct a prefix tree. It considers prefixes of trace variants of increasing lengths and obfuscates their occurrence counts using the Laplace mechanism [36]. Due to the exponential growth of the set of possible prefixes for a set of activities, infrequent prefixes are pruned to achieve an acceptable runtime of the algorithm. The privacy of query handled by this technique is guaranteed through differential privacy.

A trace-variant query τ that returns the actual frequency distribution, in general, cannot be expected to satisfy differential privacy. Hence, one relies on probabilistic queries $\hat{\tau}$ that approximate the true distribution, while satisfying the privacy guarantee. Therefore, we define differential privacy in the context of trace-variant queries as follows:

Definition 1 (Differential Privacy for Trace-Variant Query)**.** *Given a probabilistic trace-variant query $\hat{\tau}$ and privacy parameter $\varepsilon \in \mathbb{R}$, query $\hat{\tau}$ provides* ε-differential privacy*, if for all neighboring pairs of event logs $L_1, L_2 \in \mathcal{L}$ and for all subsets of possible trace-variant distributions, $\Theta' \subseteq \Theta$, it holds that:*

$$Pr[\hat{\tau}(L_1) \in \Theta'] \leq e^{\varepsilon} \cdot Pr[\hat{\tau}(L_2) \in \Theta']$$

with Θ being limited to distributions over activity sequences of a length of 2.

In the context of our work, this implies that a trace-variant query τ is said to preserve differential privacy, if the trace-variant distribution returned by query $\tau(L)$ does not significantly differ from the distribution returned by a query over a *neighboring* log, i.e., a log that contains one additional trace, $\tau(L \cup \{\xi\})$ or misses a certain trace $\tau(L \setminus \{\xi\})$, for any trace $\xi \in \mathcal{E}^*$.

It is important to note that a directly-follows query $\delta(L)$ is a special case of a trace-variant query $\tau(L)$, in the sense that a trace-variant query is counting sequences of varying length in a log L and a directly-follows-query $\delta(L)$ counts sequences of fixed length two. Therefore, the differential privacy definition for a trace-variant query $\tau(L)$ can be also applied to a directly-follows-query $\delta(L)$. Note that, for a directly-follows query, *neighboring* logs are characterized by the presence of a single pair of events that affects the directly-follows relation for one pair of activities. However, we can adopt the above definition based on the difference by a single trace in the absence of repetitive structures in the traces, since anonymization of different pairs of activities is independent of each other (known as the parallel composition rule of differential privacy [136]).

The query developed by Mannhardt et al. [92] uses the Laplace Mechanism to achieve differential privacy. When used to insert noise into a trace-variant distribution, the Laplace mechanism has considerable drawbacks. That is, the probability for a certain anonymized trace-variant distribution to be returned only depends on the syntactic distance of this distribution to the actual one. Yet, this ignores that certain distributions are less desirable than others, even when they are syntactically just as different. This can lead to several problems:

Behavior insertion problem: The Laplacian mechanism introduces noise into a trace-variant distribution in a fully random manner. Any new trace variant is considered to be equally suitable or problematic, respectively. Depending on the underlying process, however, some trace variants may easily be identified as manipulated ones.

For instance, the third trace variant in Table 6.1b contains a repetition of the *Triage* activity. Knowing that this activity is performed exactly once, an adversary could identify the respective variants as noise.

Similarly, although all traces in the original log start with the prefix $\langle$*Register*, *Triage*$\rangle$ and end with a *Release* activity, the aforementioned variant in Table 6.1b violates these patterns. Even without detailed knowledge about the process, an adversary immediately identifies this variant as artificial behavior and omits it during an attack, which effectively reduces the privacy guarantee associated with the published query result.

Behavior removal problem: The pruning strategies employed when anonymizing a trace-variant distribution also lead to the removal of behavior.

In our example, the third, fourth, and fifth variants of Table 6.1a do not appear in Table 6.1b, i.e., they are assigned a count of zero. Since pruning is applied in the construction of the prefix tree, it may have far reaching consequences: Assigning the prefix $\langle$*Register*, *Triage*, *Consul*.$\rangle$ an occurrence frequency below the pruning threshold implies that *none* of the variants with this prefix will appear in the resulting distribution. In the worst case, this effect may materialize for the prefix $\langle$*Register*, *Triage*$\rangle$ in our example, which, arguably, would render the result useless for most process analyses.

Result size problem: The noise insertion through a prefix-tree can lead to results that differ drastically in terms of absolute properties from the original log. This is particularly true for the number of traces reported by the query. Due to the anonymization the number of traces in the query result might differ by orders of magnitude from the number of traces in the original log. This limits the analytical information that can be gained from the anonymized trace-variant distribution.

In the remainder of this chapter, we introduce two approaches, SaCoFa and SaPa that

address these issues. Thereby, we provide a significant improvement with respect to the state-of-the-art approach, also called the Laplace Mechanism.

6.2 Semantics-aware Control-flow Anonymization

This section introduces SaCoFa (semantics-aware control-flow anonymization) as an approach to retrieve the anonymized behavior of an event log. Section 6.2.1 presents the general algorithm based on the exponential mechanism. Section 6.2.2 then defines the score function that SaCoFa employs for incorporating a process' semantics. Finally, Section 6.2.3 discusses pruning strategies for SaCoFa, and how the score function can decrease their negative effects.

6.2.1 The SaCoFa Algorithm

The idea of the SaCoFa algorithm is to construct a prefix tree of trace variants through stepwise expansion, where each step adds an activity or a dedicated end symbol to a branch in the tree. During this construction, prefixes are evaluated based on a *score function*, which reflects their compliance with the process' semantics, as captured in the original event log. Specifically, prefixes are categorized as *harmful* or *harmless*, depending on whether they violate semantic constraints and hence, threaten the utility of a trace-variant distribution.

While harmless prefixes are always added to the tree, some harmful prefixes typically also need to be incorporated, to achieve differential privacy. To this end, we leverage the exponential mechanism, which incorporates a score function to assign lower probabilities to prefixes that induce a stronger violation of a process' semantics. Hence, we are able to nudge the expansion of the tree to prefixes that are less harmful. In any case, all prefixes added to the tree are assigned noisy counts, i.e. taking the original frequency of a prefix within a log and adding a randomly drawn (possibly negative) amount from a Laplace distribution. To cope with the exponential growth of the prefix tree, we also prune the tree based on these noisy counts in each step of its expansion.

In Algorithm 1, we provide the pseudo-code for our algorithm. It takes as input an event log L and several parameters: the strength of the desired privacy guarantee ε, an upper bound on the trace-variant length l, and a pruning parameter p (or two pruning parameters $p_{harmless}$ and $p_{harmful}$, as detailed later). It returns $\tau'(L)$, i.e., an anonymized trace-variant distribution.

First, the algorithm initializes the prefix tree, represented as an empty set of prefixes Ξ (line 1). Next, the trace-variant distribution θ' and the current prefix length n are initialized

(lines 2-3). Then, the prefix tree is iteratively expanded. This expansion will eventually terminate once n reaches the maximal prefix length l (line 4).

Candidate generation

For each $n \leq l$, we expand the current tree by first generating a set of candidate prefixes. To obtain these candidates, we select each prefix $v \in \Xi$ that is maximal, i.e. for which $|v| = n-1$, and that has not yet been ended, i.e. $v(|v|) \neq \perp$ (line 6). Note that the first iteration is conducted for an empty prefix $v = \langle\rangle$. Then, for each $a \in \mathcal{A}(L) \cup \perp$, i.e., for any activity or the end symbol $\perp$, we generate a new candidate by appending a to v and add it to the candidate set C (lines 7-8).

As an example, lets us assume that the current tree Ξ consists of one prefix $v = \langle \mathit{Register} \rangle$. Then, the potential candidates C would be a concatenation of this prefix and each of the activities, e.g., $\langle \mathit{Register}, \mathit{Triage} \rangle$ or $\langle \mathit{Register}, \mathit{Register} \rangle$.

Tree expansion

The candidate prefixes in C are evaluated with a *score function* to classify them as harmless (C_{expand}) or harmful (C_{harm}) (lines 9-10). The definition of the score function depends on the incorporated notion of a process' semantics and will be discussed in Section 6.2.2. Here, we assume the score function to be applicable to prefixes, while the actual scoring refers to a distribution over prefixes with their frequencies all set to one.

Employing the exponential mechanism, we determine which of the harmful prefixes to add to the tree by random selection (line 11). Then, the selected harmful prefixes, together with the harmless ones, expand the prefix tree (line 12). Each of these prefixes is assigned a noisy count, based on its number of occurrences in the original log and added random noise (line 13). The Laplace noise is drawn from a Laplace distribution, scaled according to the privacy parameter ε, i.e., $\mathit{Lap}(\frac{1}{\varepsilon})$ (as discussed in Section 2.2.3). Here, we enforce that the noisy count is positive, since the decision to include the prefix has already been taken as part of the exponential mechanism.

To illustrate, reconsider the candidates from above. Prefix $\langle \mathit{Register}, \mathit{Triage} \rangle$ appears 49 times in event log L, while $\langle \mathit{Register}, \mathit{Register} \rangle$ never appears in it. To both prefixes, randomly drawn values from the Laplace distribution are added, so that we end up with new, noisy counts. For instance, this may yields counts of 44 for $\langle \mathit{Register}, \mathit{Triage} \rangle$ (noise of -5) and 3 for $\langle \mathit{Register}, \mathit{Register} \rangle$ (noise of +3).

Algorithm 1: The SaCoFa Algorithm

input : L, an event log; ε, the privacy parameter; l, the max. prefix length; p ($p_{harmless}, p_{harmful}$), the pruning parameter(s).
output: the result of $\tau'(L)$, an anonymized trace-variant distribution.

1 $\Xi \leftarrow \{\langle\rangle\}$; /* Initialize the prefix tree with an empty prefix */
2 $\theta' \leftarrow \emptyset$; /* Initialize the trace-variant distribution */
3 $n \leftarrow 1$; /* Initialize the current prefix length */
4 **while** $n \leq l$ **do** /* Consider prefixes up to length l */
5 $C \leftarrow \emptyset$; /* Initialize candidate set */
/* Select candidate prefixes to expand */
6 **foreach** $v \in \Xi \wedge |v| = n-1 \wedge v(|v|) \neq \perp$ **do**
/* For each possible activity */
7 **foreach** $a \in \mathcal{A}(L) \cup \{\perp\}$ **do**
/* Add expanded prefix to candidate set */
8 $C \leftarrow C \cup \{v.\langle a \rangle\}$;
/* Determine harmless prefix candidates */
9 $C_{expand} \leftarrow \{c \in C \mid \mathbf{score}(L, c) = 1\}$;
/* Determine harmful prefix candidates */
10 $C_{harm} \leftarrow C \setminus C_{expand}$;
/* Select prefixes; harmful prefix candidates are selected using the exp. mechanism */
11 $C_{expand} \leftarrow C_{expand} \cup Exp(C_{harm}, \mathbf{score}(L, .), \varepsilon)$;
/* Assign positive noisy count to prefixes */
12 **foreach** $v \in C_{expand}$ **do**
13 $\theta'(v) \leftarrow [\tau(L, v) + Lap(\frac{1}{\varepsilon})]_{>0}$;
14 $\Xi \leftarrow \mathbf{prune}(\Xi, \theta, p, C_{harm})$; /* Prune prefix tree */
15 $n \leftarrow n + 1$; /* Increase current prefix length */
/* Return the distribution over all prefixes that are complete or of length l */
16 **return** $\theta'|\{v \in \Theta \mid v(|v|) = \perp \vee |v| = l\}$

Tree pruning

Following the prefix tree expansion, we prune it based on the noisy counts assigned to trace variants (line 14). A simple pruning strategy removes all prefixes from the tree, for which the noisy count is below a threshold set by parameter p. However, as we will discuss in Section 6.2.3, pruning may treat harmful and harmless prefixes differently (using two thresholds, $p_{harmless}$ and $p_{harmful}$). In general, we also favor pruning of harmful prefixes to avoid the removal of prefixes that conform to the semantics of the process at hand.

Let us revisit our example and let us assume the pruning parameter was set to $p = 5$. We also assume no difference is made between harmful and harmless prefixes. In this case, the noisy count for the prefix $\langle Register, Triage \rangle$ with 44 is higher than the pruning parameter

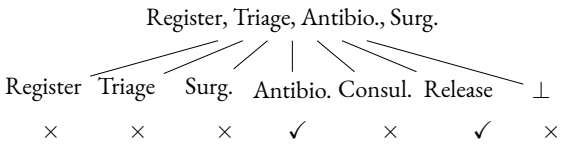


Figure 6.2: Example of potential prefix extensions. ✓ and ×, indicate harmless and harmful prefix extensions, respectively.

and the prefix is kept inside the prefix tree Ξ. Therefore, this prefix will be considered within the further extension of the prefix tree. On the other hand, for the prefix $\langle \mathit{Register}, \mathit{Register} \rangle$, we have a noisy count of 3 and the prefix is removed from the prefix tree. Prefixes that would be based on $\langle \mathit{Register}, \mathit{Register} \rangle$ will also no longer be considered in the remaining run of SaCoFa.

Result construction

Finally, the resulting trace-variant distribution is derived and returned (line 16). To this end, the counts of all prefixes that end with the symbol $\perp$ or that have a length of l are considered. Intuitively, prefixes of length l may represent variants of traces that have not yet finished execution.

6.2.2 A Semantics-aware Score Function

SaCoFa uses a score function to assess the utility loss associated with a prefix based on the process behavior included in the original log L. This function is employed to distinguish harmless prefixes (C_{expand}) from harmful ones (C_{harm}) (line 9) and in the exponential mechanism (line 11). As such, the definition of the score function denotes a design choice that enables us to incorporate different notions of a process' semantics in the anonymization.

To exemplify this design choice, we propose a function that is based on a generalization of the behavior in the original log. Specifically, we consider a behavioral abstraction that was proposed in the context of the *behavioral appropriateness* measure [130]. This behavioral abstraction defines rules between pairs of activities, reflecting their order and co-occurrence in a log. Specifically, given $a_1, a_2 \in \mathcal{A}(L)$, the rules capture if a_1 will *always*, *never*, or *sometimes* follow (or precede) an activity a_2, not necessarily directly. As such, the set of rules encodes hidden business logic, derived from a log without manual intervention.

We instantiate two score functions based on these rules, one binary and one continuous. The binary instantiation classifies all prefixes that violate at least one rule as harmful, and all

other prefixes as harmless. In contrast, the continuous instantiation aggregates the number of rule violations to quantify the harmfulness of a prefix, thus assessing the severity of the violations. Since the sensitivity of the exponential mechanism considers the maximum impact one trace may have on the score function, this degree of harmfulness needs to be limited by a user-defined upper bound.

For illustration purposes, consider the example given in Fig. 6.2, which depicts the expansion of prefix $\langle \textit{Register}, \textit{Triage}, \textit{Antibio.}, \textit{Surg.} \rangle$, based on the example from Table 6.1. In the original log, activity *Surg.* is always followed by activity *Release*, may be followed by activity *Antibio.*, and is never followed by the remaining activities. Respecting these behavioral rules, expansions based on the former two activities are considered harmless, while those in the latter are categorized as harmful.

As mentioned above, the score function may also be defined based on other behavioral models. In particular, it may be grounded in other sets of behavioral rules, such as those presented in [172, 111], which are then instantiated for the original log to capture the semantics of the underlying process. Moreover, rules may also originate from other sources, such as textual documents [147]. However, deriving the rules from the original log ensures that trace variants in the original log are more likely to be preserved.

6.2.3 Semantics-aware Pruning

To achieve differential privacy, the number of prefixes to be considered during prefix expansion grows exponentially in the prefix length. Consequently, we prune infrequent trace variants to achieve tractability, as detailed below.

The need for generalization

Pruning comes with the risk of removing prefixes (and thus trace variants) that are common in the original log, which subsequently reduces the utility of the anonymized trace-variant distribution. However, unlike log anonymization with the Laplace mechanism, SaCoFa supports a differentiation between prefixes that are harmful and harmless for an anonymized distribution. Therefore, we can limit pruning only to harmful prefixes. This way, the overall number of pruned prefixes is reduced, but harmless prefixes are always preserved, even when their noisy count is below the pruning parameter p.

A lower number of pruned prefixes, in general, also reduces privacy degradation. However, when pruning solely harmful prefixes, there is a risk to violate the required differential

privacy guarantee. That is, if harmful prefixes are characterized based on their absence in the original log, the following may happen: For two neighboring event logs, that differ by a trace of a variant that appears only in one of the logs, the anonymized variant distributions may enable the identification of the respective trace. To avoid such situations, we employ a pruning strategy that incorporates behavioral generalization.

Rule-based pruning

By employing the abstraction underlying the behavioral appropriateness measure to identify harmful prefixes for pruning, we avoid to reveal the difference between two neighboring event logs. Due to the implied behavioral generalization, a trace representing a difference between two logs may also induce a change in the respective rule sets. The changed rules potentially allow for more behavior, i.e., they increase the set of harmless prefixes. Hence, the anonymized trace-variant distributions of neighboring logs may differ by *multiple* trace variants, instead of just a single one.

For illustration, consider a log L_1 containing only traces that represent variants from Table 6.1a. Let $L_2 = L_1 \cup \{\xi\}$ be a neighboring log, where ξ is a trace with the control-flow: $\langle Register, Antibio., Release \rangle$. Comparing the rule sets of both logs, trace ξ adds the rule that *Register* is sometimes followed by *Antibio.* Hence, the SaCoFa algorithm would consider the prefix $\langle Register, Antibio. \rangle$ as harmless when anonymizing L_2, whereas it would be harmful regarding L_1. For L_2, further prefixes would then be derived and considered as harmless, e.g., $\langle Register, Antibio., Release \rangle$ and $\langle Register, Antibio., Surg., Release \rangle$. Thus, the distributions derived for the logs will differ by more than one trace variant.

Therefore, pruning only harmful prefixes requires that a single trace either leads to multiple trace variants to be considered as harmless, or none at all. In practice, this may not be the case, which is why we relax the pruning strategy, as follows. We introduce $p_{harmless}$ and $p_{harmful}$ as separate pruning thresholds for harmless and harmful prefixes, respectively. By setting $1 < p_{harmless} < p_{harmful}$, we favor pruning of harmful prefixes. Yet, by pruning also some harmless prefixes, we ensure that information on the existence of a single trace variant is not disclosed, even if the above requirement is not met. Also, the two aforementioned extreme scenarios could be configured accordingly, i.e., pruning only harmful traces ($p_{harmless} = 1$ and $p_{harmful} > 1$) or pruning all prefixes that introduce new behavior ($p_{harmless} = 1$ and $p_{harmful} = \infty$). It is important to note that pruning is always applied to the noisy counts, as inserted during the *Tree expansion* step.

6.3 Directly-follows-based Control-flow Anonymization

In this section, we complement the previously proposed approach with an algorithm that aims to overcome the issues inherent to control-flow anonymization using prefix tree construction. That is, we present semantics-aware play-out-based anonymization, short SaPa, as an approach that achieves a better preservation of absolute properties of a trace-variant distribution and avoids the need to configure parameters that have a large impact on the result, but are difficult to set in practice.

Our idea is to construct a trace-variant distribution based on a directly-follows query and a play-out procedure for the obtained result. In Section 6.3.1, we first outline the general idea behind SaPa and explain why this approach avoids the aforementioned issues. Afterwards, in Section 6.3.2, we lift the idea of semantics-aware anonymization to the anonymization of the directly-follows query, allowing us to instantiate SaPa in a semantics-aware manner as well.

6.3.1 Generating Trace Variants based on Play-out

A directly-follows distribution, capturing how often events corresponding to certain activity pairs directly follow each other in traces, denotes a certain representation of the behavior present in a log. Through play-out, this representation can be used to simulate the control-flow of the respective process, which is achieved through the step-wise concatenation of directly-follows relations to construct traces. The privacy rationale of our approach can therefore be summarized as follows: If the directly-follows distribution itself is protected by differential privacy, the result of the play-out will then also be protected by differential privacy, given that no additional information is incorporated. The latter means that the play-out is based solely on the privatized directly-follows distribution.

Following this line, an anonymized directly-follows distribution can be used to generate an anonymized trace-variant distribution. By generating a trace-variant distribution from the directly-follows distribution, we are more likely to create a result that has similar absolute properties as the result computed over the original log, i.e., the overall number of traces (and also their lengths) can be expected to be relatively close. The reason being that we consider directly-follows distributions that contain dedicated symbols to indicate the start ($\top$) and end ($\bot$) of a trace, or trace variant, respectively. Then, the number of generated traces depends primarily on the frequency of these start and end symbols. Since these frequencies are constructed from several noisy counts, for which the average of the randomly drawn noise will be close to zero, the frequencies can be expected to be relatively stable.

Algorithm 2: Play-out Algorithm

input : L, an event log.
output : the result of $\tau'(L)$, an anonymized trace-variant distribution.

1 $\theta' \leftarrow \emptyset$; /* Initialize the trace-variant distribution */
2 $dfg' \leftarrow \hat{\partial}(L)$; /* Generate ε-differentially private directly-follows distribution */
3 **while** *containsTrace*(dfg') **do** /* Checks if there is still a trace */
4 $\xi \leftarrow \langle \top \rangle$; /* Initialize trace with trace start event */
/* As long as last element is not an end of trace and it is not empty */
5 **while** $\xi_{|\xi|} \neq \perp \vee \xi \neq \langle\rangle$ **do**
/* Select next activity based on random choice from the df-distribution */
6 $a \leftarrow$ *pickNextActivity*$(dfg', \xi_{|\xi|})$;
/* Check if a next activity exists and the trace can be continued */
7 **if** $a \neq \varnothing$ **then**
/* Decrease DFG count by used directly-follows relation */
8 $dfg'(\langle \xi_{|\xi|}, a \rangle) \leftarrow dfg'(\langle \xi_{|\xi|}, a \rangle) - 1$;
9 $\xi \leftarrow \xi.\langle a \rangle$; /* Adding a to ξ to continue the trace */
10 **else**
/* All df-relation entries to the last element of ξ are removed */
11 **foreach** $a' \in \mathcal{A} \cup \top$ **do**
12 $dfg'(\langle a', \xi_{|\xi|} \rangle) \leftarrow 0$
13 $\xi \leftarrow \langle \xi_1, \ldots, \xi_{|\xi|-1} \rangle$; /* The last element of ξ is removed */
/* Trace variant distribution is updated by including the current trace */
14 **if** $\xi \neq \langle\rangle$ **then** $\theta'(\xi) \leftarrow \begin{cases} \theta'(\xi) + 1 & \text{if } \xi \in \text{dom}(\theta') \\ 1 & \text{otherwise.} \end{cases}$;
/* Return anonymized trace variant distribution */
15 **return** θ'

In Algorithm 2, we outline the SaPa approach, which uses an anonymized directly-follows distribution as a basis for the generation of an anonymized trace-variant distribution. In principle, the algorithm only requires an event log L as input, though, depending on the mechanism used to anonymize the directly-follows distribution, further inputs may be necessary.

Anonymization of the directly-follows distribution

The first step of our approach is to anonymize the directly-follows distribution. Our algorithm is independent of the choice for a specific procedure to anonymized the directly-follows distribution.

As suggested in the literature, a common choice would be a procedure based on the Laplace mechanism [92], configured by the privacy parameter ε. We later propose an alternative

procedure that accounts for a process' semantics (Section 6.3.2). In any case, it is important to note that, contrary to the approaches that anonymize trace-variants directly, we here anonymize each entry of the directly-follows relation independently.

In Algorithm 2, we first initialize an empty trace-variant distribution (line 1), before retrieving the anonymized directly-follows distribution with a privacy-preserving directly-follows query $\hat{\partial}$ (line 2).

We provide an example of an anonymized directly-follows distribution in Table 6.2. The example serves illustrative purposes and is, therefore, less noisy than what would be expected in reality to increase readability.

Play-out of the anonymized directly-follows distribution

Next, traces will be generated from the directly-follows distribution, for as long as a trace can be constructed from the start symbol ($\top$) to the end symbol ($\bot$) based on the entries in the directly-follows distribution (line 3). Here, the respective entries in the directly-follows distribution need to be subsequent, i.e., the target activity of one entry needs to be equivalent to the source of the next entry, and their counts in the distribution need to be larger than zero. If such a construction would be generally possible, beginning with a start symbol ($\top$) (line 4), we try to assemble a trace activity by activity until reaching the end symbol ($\bot$) (line 5). Once this happens, the now completed trace is included in the trace-variant distribution (line 14).

The actual expansion works as follows. The trace is extended by appending a randomly selected, directly-following activity from the anonymized directly-follows distribution (line 6). If such an activity could be found ($\varnothing$ denotes that this was not the case) (line 7), the count of the respective entry in the directly-follows distribution is reduced by one (line 8) and the trace is expanded accordingly (line 9).

Let us turn to our example in Table 6.2 to better understand these steps. In the example, a trace could be started with either the activity *Register* or *Triage*. Assume that we start a trace with *Triage*. As the next step, we could extend the trace with either *Surg.* or *Antibio.*. If the trace is extended with *Surg.*, it is only possible to continue to *Release*. From there, we finally reach the trace end $\bot$ and we would have constructed a trace $\langle \textit{Triage}, \textit{Surg.}, \textit{Release} \rangle$. We now would decrease all counts that let to that trace and restart with $\top$.

If no activity could be found for expansion of the trace, we remove all entries of the directly-follows distribution that lead to this activity (line 12). Furthermore, aiming for completion of the current prefix, we adopt a backtracking procedure. That is, we remove the last activity (line 13), and continue with the algorithm. Should it turn out to be impossible to finish con-

Table 6.2: Example of a anonymized Directly-follows Distributions

↓ follows →	⊤	Register	Triage	Surg.	Antibio.	Consul	Release	⊥
⊤	0	0	0	0	0	0	0	0
Register	44	..	...	...	0	...	...	0
Triage	3	52	...	..	0	...	...	0
Surg.	0	...	30	...	0	...	...	0
Antibio.	0	0	15	0	0	0	0	0
Consul.	0	...	..	..	0	...	...	0
Release	0	...	..	16	0	..	...	0
⊥	0	..	...	...	0	..	51	..

struction of the respective trace, backtracking will yield the empty trace, which is discarded entirely.

Considering our example, if we would have step-wise created a prefix $\langle$*Register*, *Triage.*, *Antibio.*$\rangle$, we would run into the issue that *Antibio.* has no outgoing directly-follows relations entries. Therefore, we would backtrack to $\langle$*Register*, *Triage.*$\rangle$ and remove all incoming directly-follows relations entries for *Antibio.*. Our trace would instead be continued with the activity *Surg.*, since it is the only other activity that can be reached from *Triage*.

Eventually, it will no longer be possible to construct any traces, from start to end, based on the remaining counts of the directly-follows distribution. In that case, the trace-variant distribution is returned (line 15).

6.3.2 Exponential Mechanism for Directly-follows-Query

In this section, we propose a semantics-aware procedure for the anonymization of a directly-follows query, based on the exponential mechanism. The exponential mechanism aims to optimize the model generated by the play-out of the resulting directly-follows distribution. Therefore, the main goals are to prevent the insertion of harmful directly-follows relations entries and to preserve entries that provide utility. Our intuition to achieve that goal is based on the idea that we want to avoid the procedure skipping parts of the process, by jumping from the start of the process to its end.

To ensure this intuition, we assess the utility of a directly-follows relation, i.e., how realistic its occurrence is, in terms of its *m-follows score*. This score measures the minimum distance that exists between pairs of activities for the traces in an event log. For instance, if, following events corresponding to an activity a_1, there are always at least two other events before an

Algorithm 3: Exponential Mechanism for the Directly-follows Query

input : L, an event log; ε, the privacy parameter; m, the m-follows threshold.
output: dfg', an anonymized directly-follows distribution.

1 $dfg' \leftarrow \emptyset$
2 $\theta \leftarrow \delta(L)$
3 $R_{util} \leftarrow \{\langle a_1, a_2\rangle \in dom(\theta) \mid \textbf{follows}-\textbf{score}(a_1, a_2) \leq m\}$
4 $R_{harm} \leftarrow \{\langle a_1, a_2\rangle \in dom(\theta) \mid \textbf{follows}-\textbf{score}(a_1, a_2) > m\}$
5 $R_{util} \leftarrow R_{util} \cup Exp(R_{harm}, \textbf{score}(L, .), \varepsilon)$
6 **foreach** $(\langle a_1, a_2\rangle) \in dom(\theta)$ **do**
7 **if** $\langle a_1, a_2\rangle \in R_{util}$ **then**
8 $dfg'(\langle a_1, a_2\rangle) \leftarrow [\theta(\langle a_1, a_2\rangle) + Lap(\frac{1}{\varepsilon})]_{>0}$
9 **return** dfg'

event of activity a_2 appears, the k-follows score from a_1 to a_2 is 3.

Given a specific value for m, this measure allows us to distinguish between directly-follows relations whose activities have a minimal distance below or equal to m and those that only occur further apart from each other. Intuitively, such latter relations should be avoided when establishing an anonymized directly-follows distribution, since it would mean that the directly-follows play-out would contain process behavior that stands out from what has been observed in the underlying event log.

For example, if there is always a considerable distance between occurrences of the *Register* (which usually appears as the first activity of a trace) and *Release* (usually the last activity) activities, an anonymization procedure should avoid placing these activities directly after each other. Otherwise, we would create traces where a patient is first admitted/registered in a hospital and, afterwards, immediately released without any treatment.

We use this intuition in conjunction with the exponential mechanism for a directly-follows query δ, as outlined in Algorithm 3.

The algorithm takes as input an event log L, a parameter m for the score function and a parameter ε to determine the privacy protection guarantee. First, the directly-follows distribution (line 1) is initialized. Next, all directly-follows relations of the original event log L are separated into utility-preserving R_{util} (line 3) and harmful R_{harm}(line 4), depending on their m-follows score being below or equal to, or above the threshold m.

Now, some randomly picked harmful relations from R_{harm} are added to those relations that will be preserved (R_{util}). Here, the counts depend on the privacy parameter ε (line 5). Afterwards, the procedure iterates through the directly-follows distribution (line 6) and all directly-follows relations captured in R_{util} are considered for further anonymization (line 7).

Noise scaled proportionally to the privacy guarantee ε is added to the counts of the considered relations. At the same time, we enforce that these relations need to have counts above 0, so they need to be present in the anonymized directly-follows query result. The anonymized directly-follows distribution is updated with the new privatized occurrence counts, and the procedure continues with the next relation (line 8). Once all directly-follows relations have been considered, the anonymized directly-follows distribution is provided as an output (line 9).

6.4 Evaluation

In this section, we evaluate our approaches to control-flow anonymization with SaCoFa and SaPa, which includes a comparison with the state of the art. We investigate the utility of the derived trace-variant distributions for process discovery and assess the ability to avoid obvious noise by exploring the frequency of anomalies. Below, we first review the used datasets (Section 6.4.1) and our experimental setup (Section 6.4.2). We then present our experimental results (Section 6.4.3).

6.4.1 Datasets

We use five real-world event logs as the basis for our experiments. Table 6.3 lists some essential properties of these event logs, illustrating also the motivation for selecting them. The event logs differ in their size and complexity:

The *BPIC 2013* log contains events from 7 different activities, the least number of activities of all investigated event logs. We consider it semi-structured with less than five cases per variant on average.

The *CoSeLoG* log is another semi-structured process that has on average around 12 cases per variant. It has the highest number of activities, 27 in total.

Hospital Billing is one of the larger logs we investigate. While its variants on average contain around 100 cases, it still has the highest number of variants of all logs under investigation.

The *Sepsis* log captures a very unstructured hospital-treatment process, containing 846 variants of which the vast majority occurred just once.

Finally, the *Traffic Fines* log contains data on a very structured process, with just 231 variants over a total of 150,370 traces. Furthermore, it is the event log with highest number of events in total with over 500,000 events.

Table 6.3: Descriptive statistics for the event logs.

Event Log	# Events	# Activities	# Cases	# Variants
BPIC 2013 [139]	6,660	7	1,487	327
CoSeLoG [16]	8,577	27	1,434	116
Hospital Billings [94]	451,359	18	100,000	1020
Sepsis [88]	15,214	16	1,050	846
Traffic Fines [27]	561,470	11	150,370	231

6.4.2 Experimental Setup

Baselines

We evaluate our approaches against the state of the art for the computation of anonymized trace-variant distributions, as presented by Mannhardt et al. [92]. It anonymizes the result of a trace-variant query using the Laplace mechanism, which is why we refer to this approach as 'Laplace'. Furthermore, we also consider a realization of SaPa using a directly-follows distribution that was anonymized with the Laplace mechanism, as also suggested by Mannhardt et al. [92]. We refer to this approach as 'DF-Laplace'.

Parameter settings

SaCoFa takes four parameters: the strength of the desired privacy guarantee ε, an upper bound on the trace-variant length l, and the pruning parameters $p_{harmful}$ and $p_{harmless}$. Per event log, we set l so that roughly 80-90% of the original trace variants are covered. For each of the employed privacy guarantees, i.e., $\varepsilon = \{1.0, 0.1, 0.01\}$, we explored pruning parameters starting at 2, 20, and 200, respectively, until a configuration was found such that the trace-variant query could be executed within several seconds. Overall, this approach resulted in the parameter settings given in Table 6.4, which we employed for our approach and, if applicable, for the baseline techniques. For experiments related to semantics-aware pruning, we only consider the two stronger privacy settings, because only here we were able to instantiate the pruning parameter for harmless prefixes with a significant difference to the regular pruning parameter.

For SaPa, we consider different values for the m-follows relation parameter, i.e., $m = \{1, 2, 3, 4, 5\}$. However, if the parameter is not mentioned explicitly, we use a default value of $m = 1$.

Table 6.4: Employed parameter settings.

Log	ε	l	$p_{harmful}$	$p_{harmless}$
BPIC 2013	1.0	15	2	
	0.1	15	20	10
	0.01	15	100	50
CoSeLoG	1.0	10	3	
	0.1	10	25	22
	0.01	10	220	200
HospitalBilling	1.0	13	5	
	0.1	13	20	15
	0.01	13	200	150
Sepsis	1.0	23	4	
	0.1	23	20	15
	0.01	23	190	150
Traffic Fines	1.0	9	2	
	0.1	9	20	15
	0.01	9	150	120

Evaluation measures

To quantify the efficacy of our work, we primarily assess the utility of process models discovered on the basis of the anonymized trace-variant distributions generated by the different techniques. Next to that, we also consider various measures that quantify characteristics of the privatized logs themselves.

To discover a model, we use the Inductive Miner Infrequent [76], a state-of-the-art technique for process discovery, which is commonly used (and selected here) because it guarantees process models that are free of deadlocks. The technique uses noise filtering to remove infrequent behavior, for which we select the default threshold of 20%. Furthermore, we evaluate our logs using the Heuristic Miner [173], another commonly used process discovery technique that enables us to explore long-term dependencies within a log.

Given such a discovered model, we use several metrics to measure its utility. First, we use the F-score in relation to the original event log, i.e., the harmonic mean of the *fitness* [13] and *precision* [100]. These metrics are the most widely-used metrics for process model utility. Also, we evaluate the *generalization* [17] of the process models discovered from the anonymized trace-variant distributions. For these measures, a score closer to 1 indicates a better result.

Beyond assessing the utility of discovered process models, we assess the size of the result by

considering the ratio of the number of traces within the anonymized result and the number of traces in the original log. Therefore, a ratio above 1 means that the result size increased through anonymization, whereas a value below 1 hints at a reduction in the number of traces.

Finally, as an additional evaluation dimension, we measure the fraction of easily-recognizable noise introduced as part of the anonymization. To this end, we apply a standard anomaly detection technique, which employs isolation forests [84], to the anonymized result. We train the model on the original log, before using it to detect anomalous traces in the anonymized result. As features in the learning process, we use a binary encoding of the activities, signaling if they are present in a trace. Moreover, we also encode the presence of directly-follows dependencies in a trace with a binary encoding.

Implementation

To conduct our experiments, we implemented SaCoFa and SaPa in Python. The source code is available on GitHub[*] under the MIT license. Furthermore, we used PM4Py's [14] implementation of the Inductive Miner, Heuristic Miner, and the evaluation measures. The implementation of the isolation forest is available in scikit-learn.[†]

Repetitions

To account for the non-deterministic nature of the algorithms, we perform 10 repetitions of all experiments. In the remainder, we report on the median, the upper quartile, and the lower quartile, using box plots.

6.4.3 Results

Process Discovery Utility

Fig. 6.3 depicts the F-scores of the process models constructed from the anonymized control-flow data derived by the different techniques.

In Fig. 6.3, we show that SaCoFa outperforms the other techniques for models generated by the Inductive Miner. However, we can see stronger benefits for SaCoFa in case of the smaller logs Sepsis and CoSeLoG. The only exception, is the small BPIC 2013 log, where no difference can be observed. For the larger logs, traffic fines and Hospital Billings, we can observe a small, but significant improvement.

[*] https://github.com/samadeusfp/SaCoFa
[†] https://scikit-learn.org/stable/

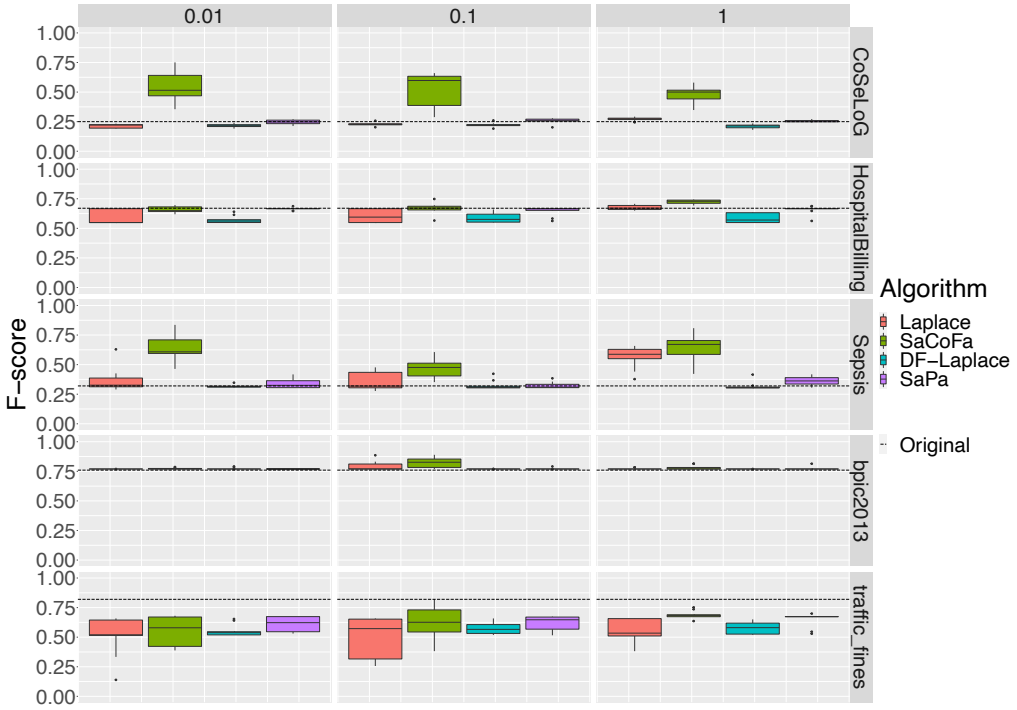


Figure 6.3: F-score of Process Models discovered with the Inductive Miner

Furthermore, our results also illustrate that, sometimes, the anonymized results lead to higher F-scores than the original log. The reason being that the Inductive Miner guarantees the generation of a fitting model, which may result in very low precision values. If an anonymized log contains less behavior, above the threshold adopted by the discovery algorithm to filter noise, the model becomes more compact. It then shows higher precision and, therefore, also a higher F-score.

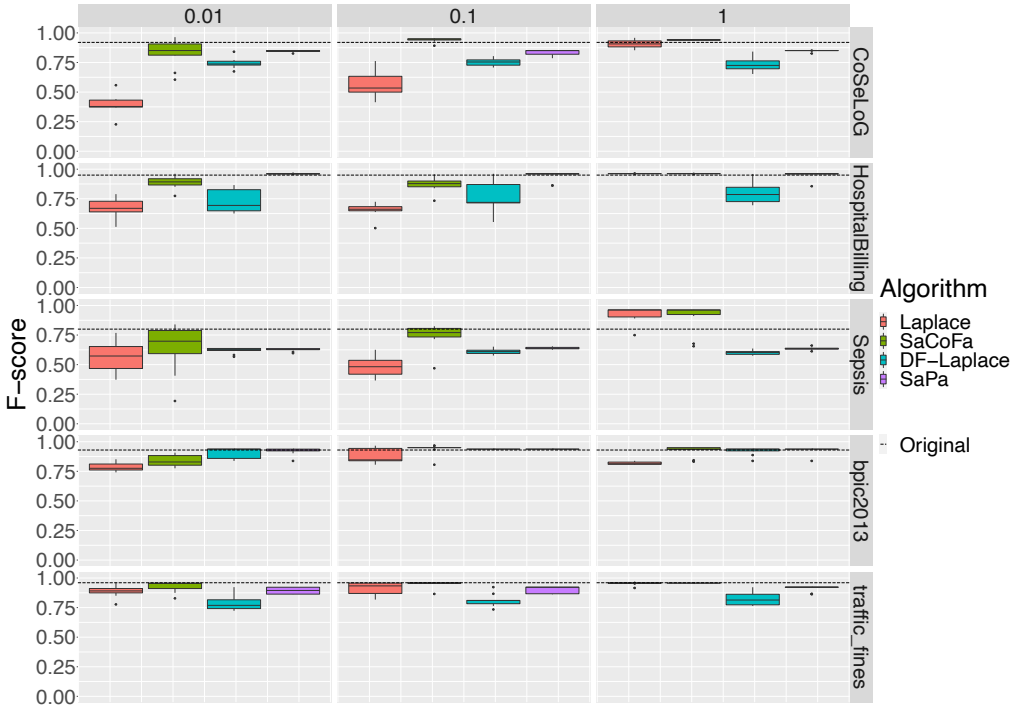
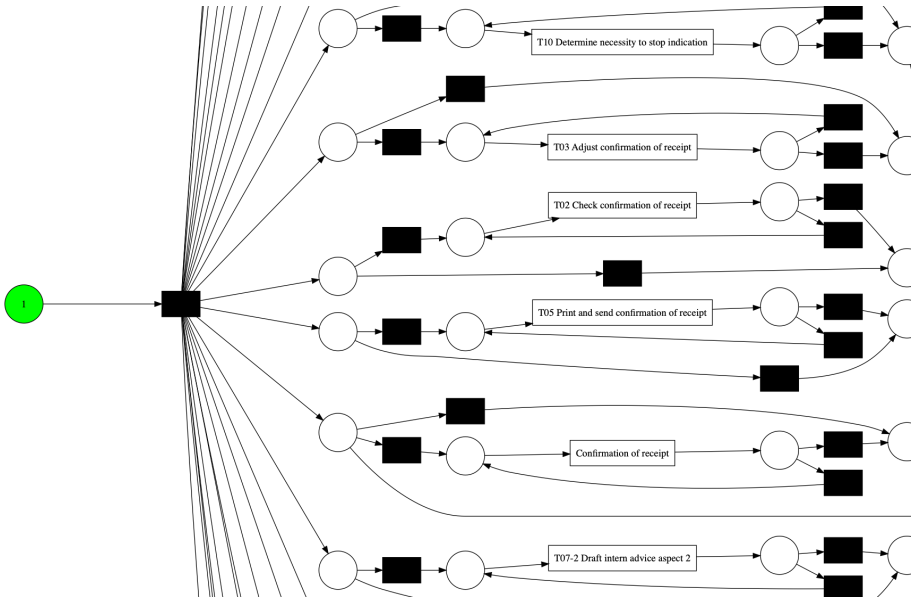


Figure 6.4: F-score of Process Models discovered with the Heuristic Miner

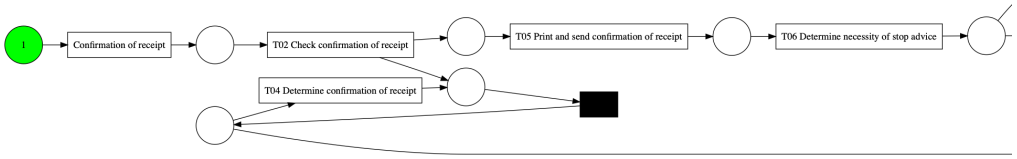
We also observe, that SaPa yields slightly better results than the DF-Laplace mechanism. However, both algorithms based on directly-follows queries achieve results similar to the direct anonymization of the trace-variant query with the Laplace mechanism and are therefore, no match with the results produced by SaCoFa.

In Fig. 6.4, we show the results retrieved from the Heuristic Miner. These results confirm that event logs anonymized using SaCoFa always outperform logs that have been anonymized by the Laplace Mechanism. The same can be said for SaPa in comparison with the DF-Laplace. However, we also see that the logs generated by SaPa outperform the logs generated by SaCoFa in case of the BPIC 2013 log under the highest privacy guarantee of $\varepsilon = 0.01$. The same is true, for the Hospital Billings log under the same privacy setting. Here, the models generated by the Inductive Miner did not show any significant difference between different anonymization strategies. One possible explanation is that the Heuristic Miner is able to find long distance dependencies, while the Inductive Miner only considers the directly-follows-graph. An approach that inserts noise on the local level introduces novel long term dependencies less systemically than a prefix-tree based approach. The latter can create a high

amount of new traces with a shared long prefix, even if they differ at the end of the trace.

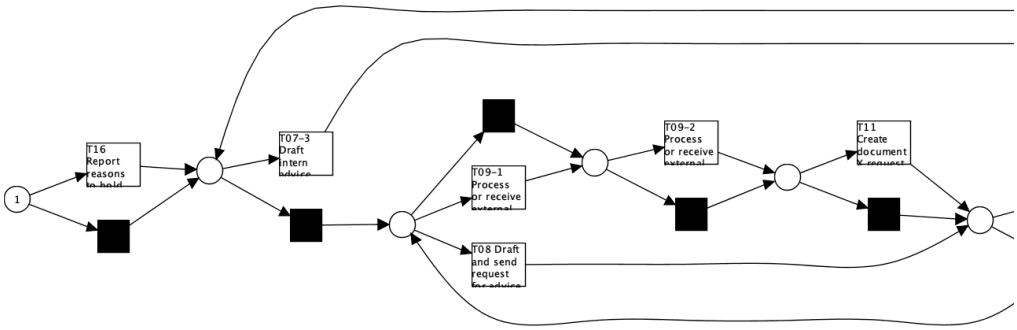


(a) Laplace baseline.

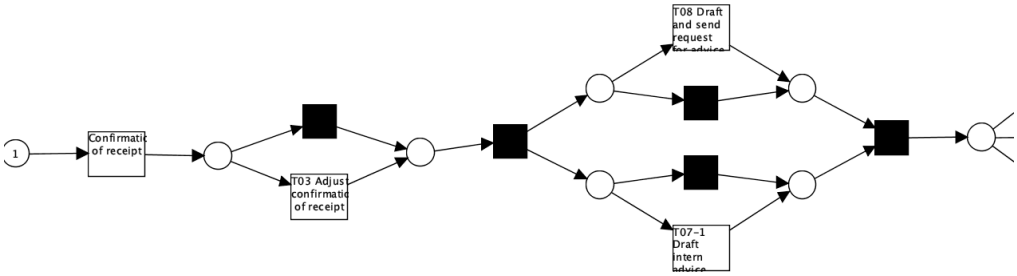


(b) SaCoFa approach.

Figure 6.5: Process models obtained for anonymized versions of CoSeLoG ($\varepsilon = 0.01$) using the Inductive Miner and Laplace Mechanism/SaCoFa.



(a) DF-Laplace approach.



(b) SaPa approach.

Figure 6.6: Process models obtained for anonymized versions of CoSeLoG ($\varepsilon = 0.01$) using the Inductive Miner and DF-Laplace/SaPa.

To further illustrate the above results, Fig. 6.5 shows excerpts of two process models obtained for the CoSeLoG process using the Laplace baseline and SaCoFa under the strongest privacy guarantee ($\varepsilon = 0.01$). We generated these process models with the Inductive Miner. Here, Fig. 6.5a shows a part of the model discovered from the result of the Laplace baseline, whereas Fig. 6.5b is based on SaCoFa. As seen, the process model generated with SaCoFa is much more structured. It starts with a sequence of activities that, notably, is also the same in the process model generated from the original event log. In contrast, the model in Fig. 6.5a is highly unstructured and strays far from the original process: nearly all activities can start a trace, be skipped, or executed multiple times. Therefore, the higher utility of the process model also comes with a better understandability of the model. This result further supports the argument that SaCoFa provides the highest utility for process discovery.

For the two approaches based of the directly-follows distribution we show the process models in Fig. 6.6. We can observe that logs generated by both DF-Laplace and SaPa lead to models that are sequential. However, we can also observe that the first activity in the model generated by DF-Laplace is not the most common start activity of the original log (*Confirmation of receipt*). Therefore, we can also conclude that the benefit in process utility of SaPa over DF-Laplace leads to process model that are actually more useful to an analyst.

Next, we turn to an assessment of the generalization of the models obtained by the Inductive Miner. As illustrated in Fig. 6.7, the models generated based on the results derived with SaCoFa are more general than the models generated by the Laplace baseline technique, i.e., they abstract more from the represented behavior. However, the observed benefits by SaCoFa are mostly minor.

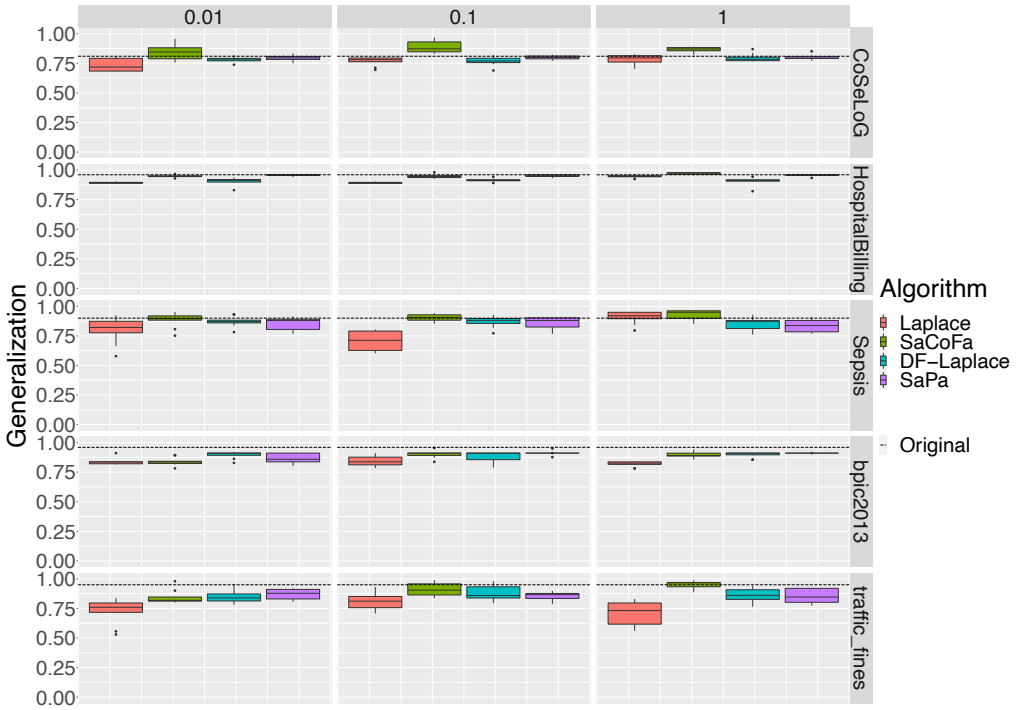


Figure 6.7: Generalization of discovered process models generated by the Inductive Miner.

We also observe slight benefits for SaPa over DF-Laplace. While we can conclude, that the exponential approaches are better than their Laplace counterparts, the comparison between SaPa and SaCoFa is less clear. Often, SaCoFa performs best, but this is for example not true

for the traffic fines log with the highest privacy guarantee of $\varepsilon = 0.01$.

When turning to the process models generated by the Heuristic Miner, shown in Fig. 6.8, we can see that for this process discovery technique SaPa provides models that generalize more than SaCoFa. However, we also are able to observe that both SaPa and SaCoFa outperform their Laplace baselines.

Overall, we can conclude that both SaCoFa and SaPa provide better process model utility than their Laplace counterparts. While SaCoFa can produce models which usually outperform all other techniques in terms of F-score the same can not be said in terms of generalization, since SaPa produces models that generalize more when considering process discovery with the Heuristic Miner.

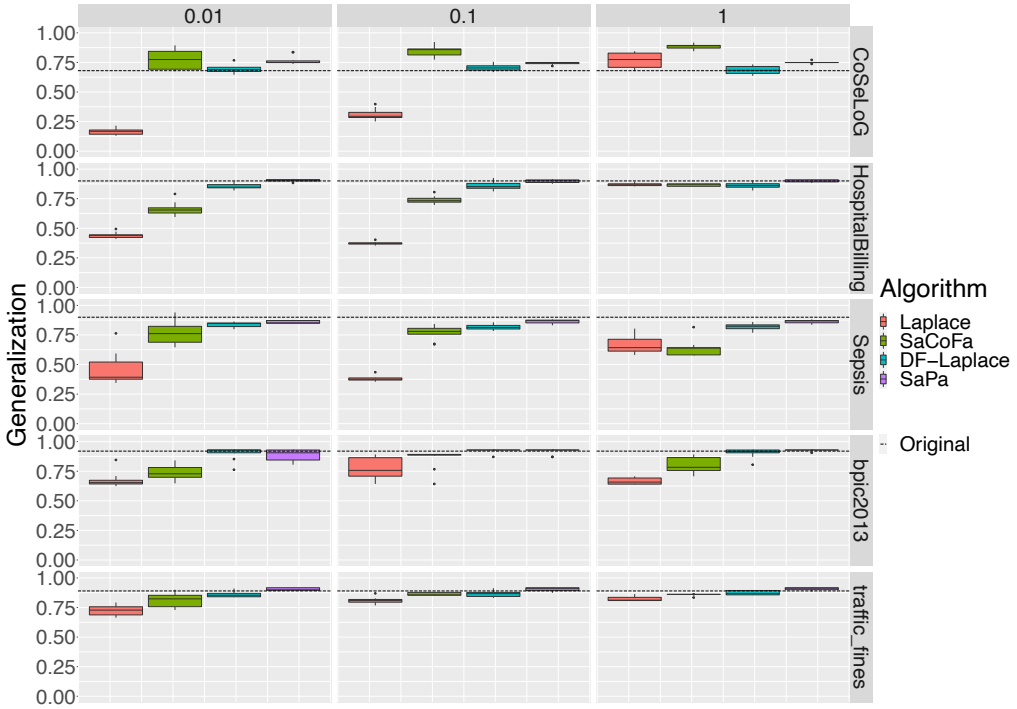


Figure 6.8: Generalization of discovered process models generated by the Heuristic Miner.

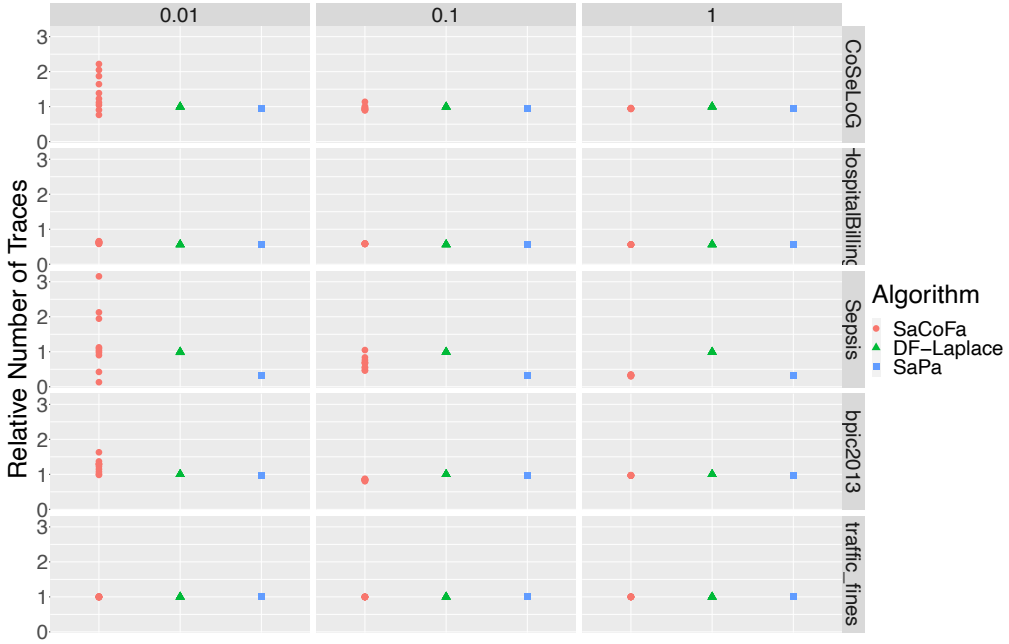


Figure 6.9: Relative size of anonymized logs compared to the original log.

Next, we turn to the number of traces in the trace-variant distributions generated by the respective techniques. In Fig. 6.9, we show the relative number of traces within the anonymized results generated by SaCoFa, SaPa, and DF-Laplace. The DF-Laplace approach generally produces distributions that have nearly the same number of traces as the original log. We can see the same result for SaPa for all cases, except the unstructured log (Sepsis). However, even for this log, the anonymized results are also consistent in their size, over several runs and ε values. Consequently, the introduced error is likely due to the play-out procedure for that specific case. On the other hand, SaCoFa produces anonymized results of very different sizes, especially for strong privacy guarantees (low ε). The SaCoFa results can contain two to three times more traces than the original log. This observation does not hold for the Traffic Fines event log, which may be attributed to the log's overall larger size.

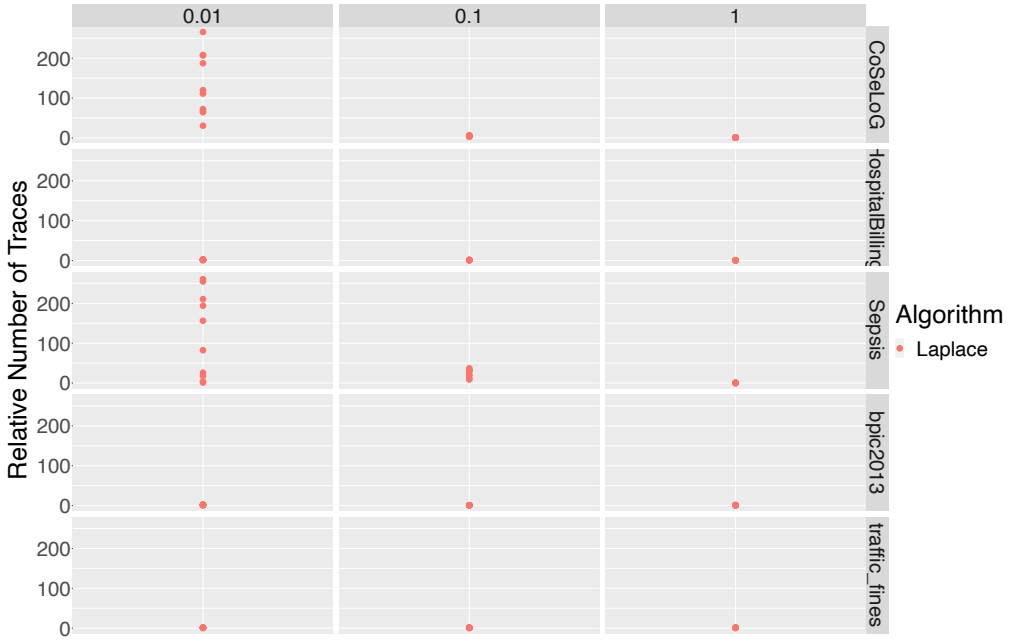


Figure 6.10: Relative size of anonymized logs compared to the original log

We present the results for the baseline that adopts the Laplace mechanism directly for the trace-variant distribution separately, in Fig. 6.10, to ensure the readability of the figures. This approach may produce results that have more than 200 times more traces than the original log, for small values of ε. Put differently, the anonymized results do not provide any reliable information about the actual size of the original log. However, the very large event logs Traffic Fines and Hospital Billings seem to be immune against this effect. The same seems to be true for BPIC 2013, the log that contains the lowest number of activities.

Based on the results for the Laplace Mechanism as well as the high variability in the results obtained with SaCoFa, we conclude that results generated by playing out an anonymized directly-follows distribution are beneficial in terms of preserving the number of traces of the log in the obtained trace-variant distribution.

Noise Insertion

Next, we turn to the presence of easily-recognizable noise. In Fig. 6.11, we show the percentage of behavior that is classified as normal behavior by the aforementioned anomaly detection technique. While all techniques achieve good results for the Traffic Fines dataset, there is a

clear trend for the other four logs: the Laplace baseline technique produces much more noise that is directly recognizable as anomalous. Therefore, the traces introduced by SaCoFa, SaPa, or DF-Laplace are more in line with the original process' behavior. However, in case of the BPIC 2013 log, SaCoFa is outperformed by both SaPa and DF-Laplace.

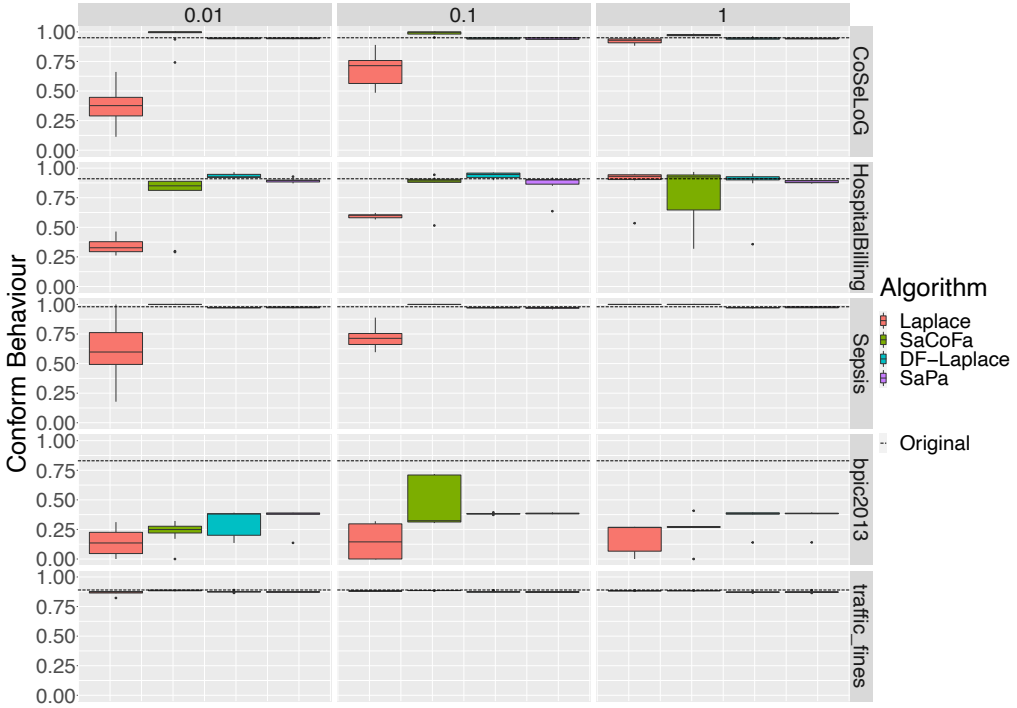


Figure 6.11: Relative frequency of normal behavior in event logs.

Internal Aspects of SaCoFa and SaPa

Finally, we investigate technical aspects of SaCoFa and SaPa. We first turn to an experiment regarding the effects of semantics-aware pruning for SaCoFa. Fig. 6.12 shows the F-score of Inductive Miner models discovered from the anonymized results obtained with and without pruning. Overall, semantics-aware pruning turns out to only be beneficial for the Traffic Fines log and the Hospital Billing log, which are the largest logs. For the less-structured logs, the F-score actually decreases in comparison to the approach without pruning. We attribute this observation to the significance of the rules used to separate harmful and harmless prefixes. Apparently, they are not always sophisticated enough to compensate for the additional variance introduced to the trace-variant distribution.

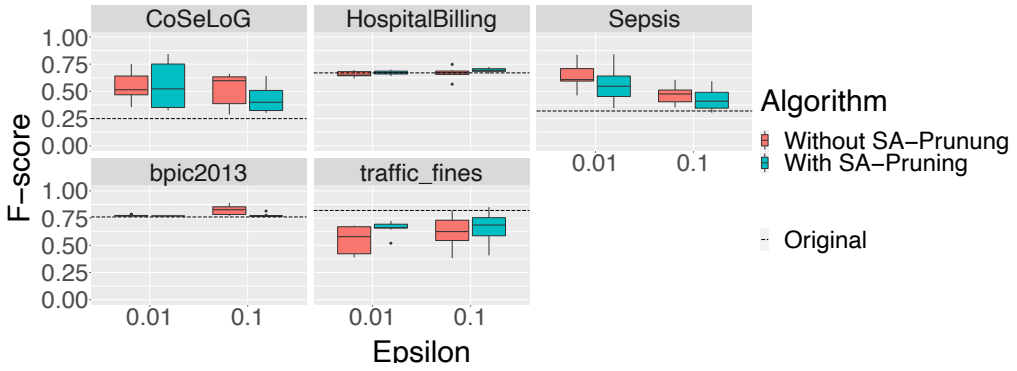


Figure 6.12: F-score for Inductive Miner configurations with and without pruning of SaCoFa.

The results for process models from the Heuristic Miner are shown in Fig. 6.13. We can not replicate the advantage of semantics-aware pruning for the Hospital Billings log. Overall, the results confirm our observation, that semantics-aware pruning with our approach does not necessary outperform a method with regular pruning.

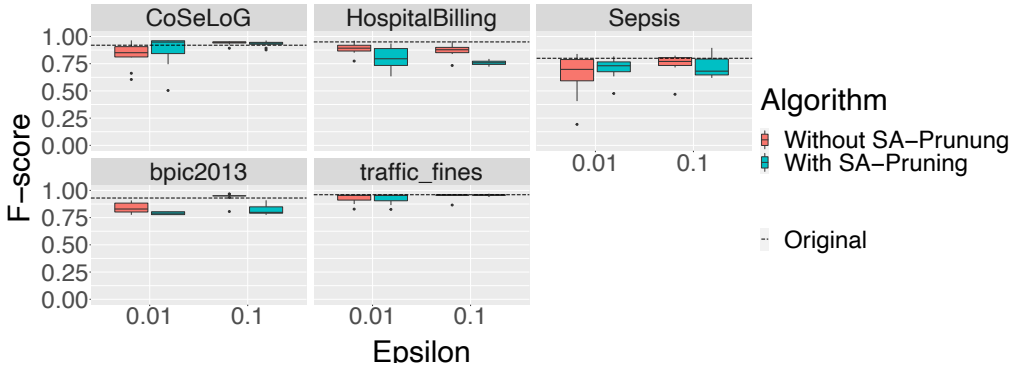


Figure 6.13: F-score for Heuristic Miner configurations with and without pruning of SaCoFa.

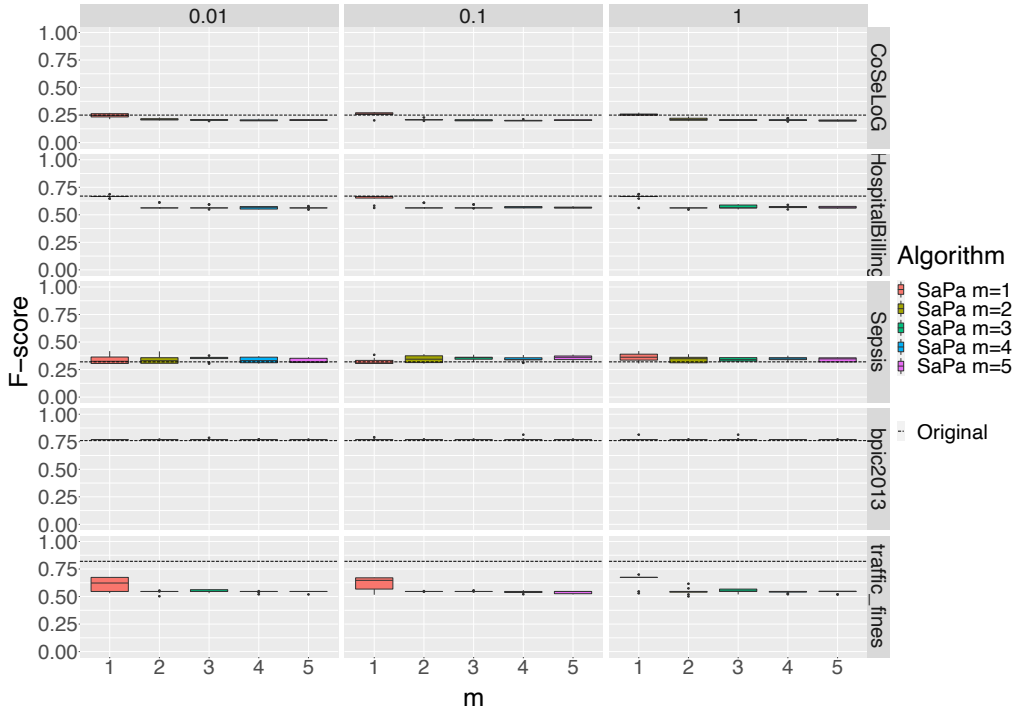


Figure 6.14: F-score for different configurations of SaPa with Inductive Miner Process Models.

For SaPa, we investigated the effects of different values for the m-follows relation threshold, as employed in the score-function of the exponential mechanism. In Fig. 6.14, the F-score of the Inductive Miner for different thresholds is shown. Overall, no general trend can be seen, but it seems that sometimes $m = 1$ produces superior results. However, in Fig. 6.15 we show the results from the same experiments, but using the Heuristic Miner. Here, higher values for m lead to lower F-scores. This result confirms our early observation that the Heuristic Miner is good at preserving long-term dependencies. Therefore, we conclude that a value of $m = 1$ is generally beneficial.

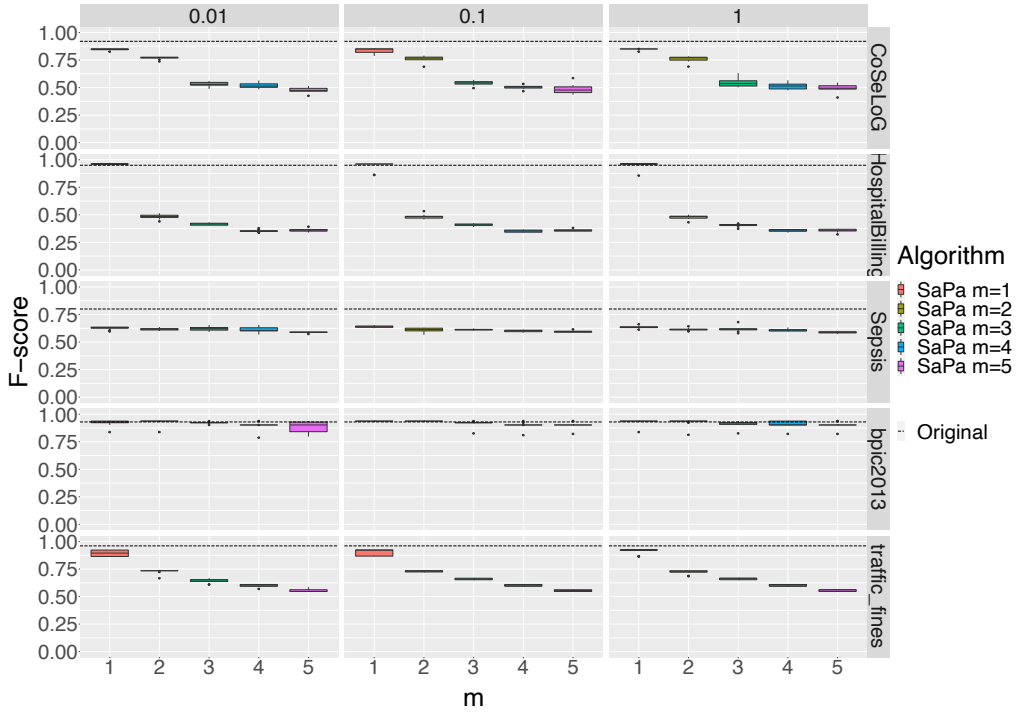


Figure 6.15: F-score for different configurations of SaPa with Heuristic Miner Process Models.

6.5 Discussion

We now turn to a discussion of different observations and characteristics of our proposed SaCoFa (Section 6.5.1) and SaPa (Section 6.5.2) approaches.

6.5.1 SaCoFa

Runtime aspects

As mentioned in Section 6.4.2, the employed parameters of SaCoFa must be selected carefully for trace-variant queries to complete in a reasonable time. Specifically, we observed that the anonymization procedure either terminated within seconds, or not all, for both SaCoFa and the Laplace trace-variant query. This reveals that there is a clear point when the prefix growth makes the trace-variant query intractable. Until now, this point is determined by step-wise altering the maximal variant length (l) and pruning parameters for a given ε. Nevertheless, we observe that SaCoFa can compute results for lower pruning thresholds faster than the Laplace

baseline. However, to ensure a fair comparison, we used the same pruning parameters for all mechanisms in our experiments.

Non-binary score functions

Beyond determining if a prefix is harmful or not, the behavioral appropriateness-based score can be used to quantify its degree of harmfulness. However, the sensitivity of the exponential mechanism depends on the maximal impact that a single case can have on the employed score function, i.e., on the function's maximal value. Therefore, if we define a score function that quantifies harmfulness in, e.g., the range $[0, 3]$, the query's sensitivity would thus be $\Delta f = 3$, instead of $\Delta f = 1$ for a binary assessment. Since the exponential mechanism inserts noise proportional to the sensitivity, such a non-binary classification leads to larger magnitudes of noise inserted. Therefore, the intended benefits obtained from quantifying harmfulness in a non-binary manner must outweigh the increased sensitivity for it to have a noticeable effect. While this did not appear to be the case in our current experiments, we believe that this approach may still be applicable for more sophisticated score functions, tailored to the specifics of the process at hand. In any case, it is important to consider this trade-off while selecting appropriate pruning parameter values.

Restrictiveness of longer prefixes

Each activity within a prefix potentially adds new rules that increase the number of harmful prefixes. Therefore, longer prefixes tend to be more restricted than shorter prefixes. This observation might be relevant, since it can create an argument for adjustments of the score-function for event logs with extremely long traces, for instance by only considering prefix expansions that violate two or more behavioral appropriateness-based rules.

Privacy Guarantee

The SaCoFa algorithm anonymizes the retrieved trace-variant distribution using noisy counts, drawn from a distribution adjusted to the differential privacy parameter ε and sensitivity of the function Δf (for trace variant queries, $\Delta f = 1$), taken as input. Therefore, data anonymized in this manner is known to be protected by ε-differential privacy [35]. However, one can argue that the actual protection may be stronger. The reason for this is that the pruning used by SaCoFa means that uncommon variants are less likely to be present in the final output of the algorithm, which reduces the difference in output between neighboring logs. Conse-

quently, the pruning part of SaCoFa might even lead to a slightly higher privacy protection than chosen by the user.

Semantics-aware Pruning

In our evaluation we did not observe a significant improvement through semantics-aware pruning. We would expect such an effect, if some useful prefixes would be included within result with semantics-aware pruning that would not be included without semantics-aware pruning. For this happen the prefix must be identified as useful by the score-function and the prefix count must be higher than the pruning parameter for useful prefix, but also lower than the pruning parameter for other prefixes. Furthermore, the additional prefix must include information that is not yet covered by the other prefixes in the anonymized trace-variant distribution. While these conditions have not been met in our data it is still possible that for other event logs or parameters a benefits could exist.

6.5.2 SaPa

Unused directly-follows relations

One consequence of the play-out algorithm of SaPa is that some directly-follows relations are not considered for the final trace-variant distribution. As shown in Section 6.4, the approach was still able to generate trace-variant distributions with a reasonable utility. However, we believe it is important to notice this inherent information loss. It may be possible to minimize this loss through more sophisticated play-out algorithms, which involves some of the considerations discussed below.

Guided play-out algorithms

It is important to notice that the privacy guarantee for SaPa was given for the anonymized directly-follows distribution. Therefore, this distribution can be processed in any way possible, assuming that no other information about the event log is added. However, if the play-out algorithm would somehow be guided by information extracted from the original data, this would subsequently intervene with the given assumption, and thus with the privacy guarantee. Consequently, the privacy of the individuals would be harmed by more than allowed for by ε. Contrary, information extracted based on the labels of the activity, would not devalue the given privacy guarantee, since these information are publicly revealed and therefore public knowledge. Therefore, for the development of play-out using additional

information, it is necessary to extract the used information only from the anonymized distribution, i.e. by heuristics that minimize the number of unused directly-follows relations, or by usage of activity labels [148].

Sensitivity of the directly-follows query

The sensitivity, i.e., the maximum impact an individual may have on the query result needs to be known in advance when applying differential privacy. For a trace-variant query, we assume this to be $\Delta f = 1$, i.e., one individual is only represented by exactly one trace. While this may not always be the case, it was shown that a strong privacy protection can still be given, without raising the sensitivity [67]. Since, even if some individuals appear multiple times the overall protection is still high. Nonetheless, determining the maximum influence one individual may have on a directly-follows distribution is more complex. Even if the assumption of one individual being represented by one trace holds, some directly-follows relations could be impacted by values far higher than 1 due to loops in the underlying process. Therefore, fixing the sensitivity to $\Delta f = 1$ might not be appropriate for processes involving a lot of loops. In such cases, it is either necessary to adjust the sensitivity accordingly or stick to SaCoFa, which by design is more robust.

Privacy Guarantee

In SaPa, the trace-variant distribution is retrieved from a differential private directly-follows distribution. Since differential private data is immune to post-processing, i.e., the guarantees remain when additional operations are applied afterwards, the final result of SaPa is therefore also protected by differential privacy [35]. Nonetheless, as discussed in the previous paragraph, setting the sensitivity Δf right for directly-follows queries is challenging and an incorrect setting can lead to a drop in the given privacy-guarantee.

6.6 Conclusion

Targeting control-flow anonymization for business processes, we introduced two approaches to answer trace-variant queries in a privacy-preserving manner. First, we introduced SaCoFa, an approach based on a prefix tree construction and the exponential mechanism. Unlike state-of-the-art techniques that leverage the Laplace mechanism and, hence, introduce random noise to achieve differential privacy, SaCoFa incorporates the semantics of the underlying process when inserting noise, achieving the same privacy guarantee. To this end, we

introduced a score function that differentiates prefixes as being harmful or harmless, which then guides the anonymization.

Second, we presented SaPa, an approach to anonymize trace-variant queries based on the play-out of anonymized directly-follows distributions. Again, we showed how to incorporate a process' semantics in the noise-insertion procedure. This is achieved through a score function that nudges towards the consideration of entries of the directly-follows relation that relate to activities that appear close to each other in the original event log. However, the main benefits of SaPa lay in its better practical applicability and the preservation of properties in the generated anonymized trace-variant distribution in terms of the number of traces. The latter aspect is of particular relevance for many analysis scenarios.

Our evaluation experiments highlight that process models generated based on control-flow behavior anonymized with SaCoFa have higher utility than those obtained with the state of the art. At the same time, they are more general and, hence, abstract better from the behavior represented in the event log. Although the utility achieved by SaPa can be lower compared to SaCoFa, SaPa generates distributions that are close to the original event log in terms of the number of traces. Moreover, we showed that SaCoFa and SaPa introduce less easily-recognizable noise in comparison to the state of the art.

In terms of limitations, our approaches assume a static number of traces corresponding to the same individual and due to privacy considerations insert new behavior into the anonymized logs. As future work, alternative metrics that can be employed as score functions for SaCoFa or SaPa can be investigated, so the mechanisms can be further tailored towards specific analytical properties. Furthermore, the development of play-out algorithms to minimize the loss of directly-follows relations through SaPa provides a potential angle to improve our work

This chapter is based on concepts and results previously published in [55], in the Proceedings of the International Conference on Business Process Management 2020.

7

PRIPEL: Privacy-preserving Event Log Publishing with Contextual Information

For many process mining use cases, it is necessary to analyze not only the control-flow, but also additional information from an event log. An example of this is multi-dimensional analysis that incorporates the impact of context on the process execution. The aim of such analysis could be the prediction of remaining waiting times for patients in an emergency room [133]. However, all techniques discussed so far are limited to the anonymization of the control-flow and neglect contextual information, such as data attributes or timestamps, thus precluding any form of process analysis that involves contextual factors.

To bridge this gap, we introduce PRIPEL, a framework for privacy-preserving event log publishing. Compared to the previously discussed techniques based on differential privacy, PRIPEL takes a fundamentally different angle and ensures privacy on the level of individual cases instead of a counting query. This way, contextual information is preserved, which enables the application of a rich set of process analysis techniques. PRIPEL utilizes the maxim of parallel composition of differential privacy. It is based on the idea of enriching a differential private trace-variant query with contextual information from the original log. Subsequently, the integrated contextual information is anonymized following the principle of local differential privacy. Notably, PRIPEL can use all known approaches of trace-variant queries. It is therefore possible to integrate all known control-flow anonymization techniques that guarantee differential privacy, based on the needs of the process analyst.

We demonstrate the feasibility of our approach through a case study in the healthcare domain. Applying PRIPEL to a real-world event log of sepsis cases from a hospital, we show that the anonymization preserves utility on the level of event-, trace-, and log-specific characteristics.

Within the remainder of the section, we start by outlining the problem in Section 7.1. Next, we introduce our approach in Section 7.2. We provide a proof-of-concept of our approach in Section 7.3. Within Section 7.4, we discuss different aspects of our algorithm. We finish our chapter with a conclusion in Section 7.5.

7.1 Problem Illustration

In Chapter 6, we discussed differentially private queries. Providing differential privacy to the result of a query is a common strategy. However, the concept of *local* differential privacy anonymizes a dataset itself in such a way that it can be published while still guaranteeing the privacy of an individual's data [68]. This is achieved by applying noise to the data, rather than applying it to the result of a query performed on the undisclosed data. The adoption of local differential privacy in industry is well-documented, being employed by, for example, SAP [69] and Google [47].

To apply this notion in the context of event logs, we define $\alpha : \mathcal{L} \to \mathcal{L}$ as an *anonymization function* that takes an event log as input and transforms it into an anonymized event log. This transformation is non-deterministic and is typically realized through a stochastic function. As before, we define $\text{img}(\alpha) \subseteq \mathcal{L}$ as the *image* of α, i.e., the set of all event logs that may be returned by α. Finally, we define two event logs $L_1, L_2 \in \mathcal{L}$ to be *neighboring*, if they differ by exactly the data of one individual. In our setting, this corresponds to one case and, hence, one trace, i.e., $|L_1 \setminus L_2| + |L_2 \setminus L_1| = 1$. Based on [68], we use the following definition for local differential privacy:

Definition 2 (Local Differential Privacy). *Given an anonymization function α and privacy parameter $\varepsilon \in \mathbb{R}$, function α provides ε-local differential privacy, if for all neighboring pairs of event logs $L_1, L_2 \in \mathcal{L}$ and all subsets $\rho_\alpha \subseteq \text{img}(\alpha)$, it holds that:*

$$Pr[\alpha(L_1) \in \rho_\alpha] \leq e^\varepsilon \times Pr[\alpha(L_2) \in \rho_\alpha]$$

where the probability is taken over the randomness introduced by the anonymization function α.

Within the remainder of this chapter we will introduce an instantiation of α. It yields the first approach that allows for the publishing of a complete anonymized event log, therefore enabling a rich set of process mining techniques.

7.2 The PRIPEL Framework

The *Pri*vacy-*P*reserving *e*vent *l*og publishing (*PRIPEL*) framework takes an event log as input and transforms it into an anonymized one that includes contextual information and guarantees ε-differential privacy. As depicted in Fig. 7.1, the PRIPEL framework consists of three main steps. Given an event log L, PRIPEL first applies a *trace-variant query* τ on L. The query returns a bag of activity sequences that ensures differential privacy from a control-flow perspective. Second, the framework constructs new traces by enriching the activity sequences obtained by $\tau(L)$ with contextual information, i.e., timestamps and attribute values, from the original log L. This is achieved in a *sequence enrichment* step, which results in a *matched* event log L_m. Finally, PRIPEL anonymizes the timestamps and attribute values of L_m individually by exploiting the maxim of parallel composition of differential privacy. The resulting event log L' then guarantees ε-differential privacy, while largely retaining the information of the original log L.

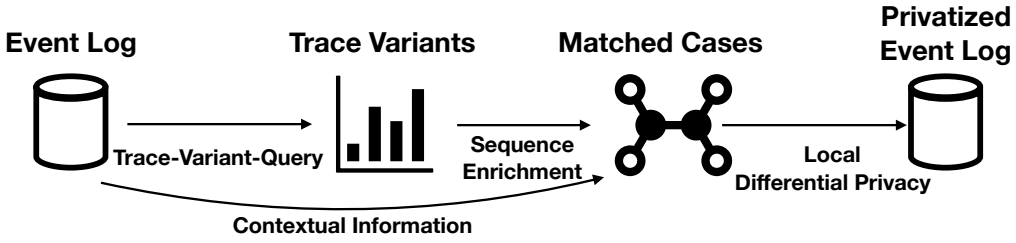


Figure 7.1: Overview of the PRIPEL Framework.

Sections 7.2.1 through 7.2.3 outline instantiations of each of these three steps. However, we note that the framework's steps can also be instantiated in a different manner, for instance by using different trace-variant queries or matching techniques. It is therefore possible to tailor PRIPEL to specific use cases, such as a setting in which traces become available in batches.

7.2.1 Trace-Variant Query

The first step of our framework targets the anonymization of an event log from a control-flow perspective. In particular, the framework applies a trace variant query, which returns a trace-variant distribution. Such a step is essential, given that even the publication of traces from an event log, i.e., with all attribute values and timestamps removed, can be sufficient to link the identity of individuals to infrequent traces [6]. For example, uncommon treatment paths may suffice to resolve the identity of a specific patient.

To avoid such privacy threats, PRIPEL only discloses traces (and their frequencies) that have been obtained by applying a query τ to L, where τ is a trace-variant query that guarantees differential privacy. Various approaches for answering a trace-variant query can be applied, including all approaches discussed in Chapter 6. It depends on the specific use-case, which approach suits best.

7.2.2 Sequence Enrichment

The second step of the framework enriches the traces-variant distribution obtained by the trace-variant query with contextual information, i.e., with timestamps and attribute values. This is achieved by establishing a trace matching between each activity sequence from $\tau(L)$ and a trace of the original log L. The latter trace determines how the trace-variant distribution is enriched with contextual information to construct a trace of the matched log L_m. Here, L_m should resemble the original log: Distributions of attribute values and timestamps, along with their correlation with trace variants in the original L shall be mirrored in the matched log L_m.

To link the trace-variant distribution in $\tau(L)$ and traces in log L, we need to match these. For this purpose, we define a matching function $g_m : B(A^*) \nrightarrow L$ that maps the bag of all activity sequences $B(A^*)$ retrieved from $\tau(L)$ to the log L. It is potentially partial and injective, i.e., it matches each activity sequence σ (note that the traces obtained from the trace-variant query are duplicated according to their frequency) to a separate trace in L, such that $g_m(\sigma_1) = g_m(\sigma_2)$ implies that $\sigma_1 = \sigma_2$ for all σ_1, σ_2 that are part of $B(A^*)$.

However, constructing such a mapping function requires to address two challenges:

(i) Since the trace variant query introduces noise, some activity sequences from $B(A^*)$ cannot be paired with traces in L that are of the exact same sequence of activity executions. Given an activity sequence $\sigma = \langle \mathit{Registration}, \mathit{Triage}, \mathit{Release} \rangle$ retrieved from $\tau(L)$ and a trace ξ with the activity sequence $\langle \mathit{Registration}, \mathit{Release} \rangle$, for example, the trace does not provide attribute values to be assigned to a *'Triage'* event. To preserve their order, the insertion of an additional event may require the timestamps of other events to be changed as well.

(ii) Since $B(A^*)$ may contain a higher trace-variant count than the amount of traces existing in the original log L , some activity sequence in $B(A^*)$ might not be matched to any trace in L, i.e., g_m is partial. Since all activity sequence in $B(A^*)$ must be retained in the construction of traces for the matched log to ensure differential privacy, also such *unmatched* sequences must be enriched with contextual information.

Given these challenges, PRIPEL incorporates three functions: (1) a matching function g_m; (2) a mechanism g_e to enrich a matched sequence σ with contextual information from trace $g_m(\sigma)$ to construct a trace for the matched log L_m; and (3) a mechanism g_u to enrich an unmatched sequence to construct a trace for L_m. In the remainder, we will show instantiations of these functions.

Matching function

The matching function g_m shall establish a mapping from $B(A^*)$ to L such that the bag of activity sequences and traces from the log are as *similar* as possible. This similarity can be quantified using a distance function. Here, we propose to use the Levensthein distance [79] to quantify the edit distance of some trace $\sigma \in B(A^*)$ and a trace $\xi_L \in L$, denoted as $\mathrm{ed}(\sigma, \xi_L)$. Using assignment optimization techniques, the matching function is instantiated, such that the total edit distance is minimized, i.e., we minimize $\sum_{\sigma \in B(A^*)} \mathrm{ed}(\sigma, g_m(\sigma))$.

Matched sequence enrichment

Given an activity sequence σ of $B(A^*)$, the sequence σ is enriched based on the context information of trace $\xi_L = g_m(\sigma)$ to create a new trace ξ_σ. The proposed procedure for this is described by Algorithm 4. To create the events for the new trace ξ_σ created based on σ, we iterate over all activities in σ, and check if there is a corresponding event e' of ξ_L. Using k_σ as the number of times we have observed activity a in the sequence σ (line 6), e' shall be the

Algorithm 4: Matched Sequence Enrichment

input : L an event log; σ an activity sequence; $\xi_L = g_m(\xi_\tau)$ the matched trace
output: A trace ξ_σ derived by enriching σ based on ξ_L.

```
1  i ← 1;
2  ξ_σ ← create new trace;
3  while 1 ≤ i ≤ |σ| do
4      e ← create new event;
5      e.a ← σ(i) ;                                   /* Assign activity to new event */
6      k_σ ← |{1 ≤ j ≤ |σ| | σ(j) = e.a}| ;    /* Count a-events in activity sequence σ */
7      k_L ← |{1 ≤ j ≤ |ξ_L| | ξ_L(j).a = e.a}| ;    /* Count a-events in original trace ξ */
8      if k_τ < k_L then
           /* Get corresponding occurrence of a                                        */
9          e' ← ξ_L(j) with ξ_L(j).a = e.a and |{1 ≤ l < j | ξ_L(l).a = e.a}| = k_σ
10         foreach d ∈ (d_e(e) \ {ts}) do
11             e.d ← e'.d ;                          /* Assign attribute values of e' to e */
12         if e'.ts > ξ_σ(|ξ_σ|).ts then
13             e.ts ← e'.ts;
14         else
15             e.ts ← derive timestamp based on Equation 7.1;
16     else
17         foreach d ∈ (d_e(e) \ {ts}) do
18             e.d ← draw random attribute value;
19         e.ts ← draw random timestamp for activity e.a ;
20     ξ_σ ← ξ_σ.⟨e⟩;
21 return ξ_σ ;                                                  /* Return new trace */
```

k_σ-th occurrence of an event in ξ_L with activity a (line 9). If such an event e' exists, we assign all its attribute values to the new event e (line 11). Subsequently, we check if the timestamp of e' occurs after the timestamp of the last event of ξ_σ (line 12). If this is the case, we assign the timestamp $e'.ts$ of the original event to event e. Otherwise, we generate a new timestamp based on the following equation, assuming that the current event is the n-th event to be added to $\xi_\sigma = \langle e_1, \ldots, e_{n-1} \rangle$:

$$e_n.ts = e_{n-1}.ts + \Delta_{e_{n-1}.a, e_n.a} \tag{7.1}$$

Here, $\Delta_{e_{n-1}.a, e_n.a}$ denotes a timestamp difference randomly drawn from the distribution of these differences in the original log. That is, the distribution is obtained by considering all pairs of subsequent events in the original traces that indicate the execution of the respective activities. If no such pairs of events appeared in the original log, we resort to the distribution of all timestamp differences of all pairs of subsequent activities of the original log.

If no corresponding event e' can be found for the newly created event e, we assign randomly drawn attribute values and a timestamp to this event (lines 18–19). We draw the attributes values from the overall distribution of each attribute d in the original log L, while timestamps are calculated according to Equation 7.1.

Unmatched sequence enrichment

For sequences in $\tau(L)$ without a matching, we assign the attribute values randomly. To handle the timestamps, we randomly draw a timestamp t_{start} for the event created for the first activity in ξ_τ, from the overall distribution of all timestamps of the first events of all traces ξ_L in the original log L. We generate the remaining timestamps based on Equation 7.1.

The runtime complexity of the whole sequence enrichment step is dominated by the assignment optimization problem, which requires $\mathcal{O}(|B(A^*)|^3)$ time [40].

7.2.3 Applying Local Differential Privacy

Next, starting with the matched log derived in the previous step, we turn to the anonymization of contextual information using local differential privacy. While the treatment of attribute values follows rather directly from existing approaches, we propose a tailored approach to handle timestamps. The runtime complexity of this step is linear in the size of the matched log L_m, i.e., we arrive at $\mathcal{O}(|L_m|)$.

Anonymizing attribute values

We differentiate between attributes of three data types: numerical, categorical, and boolean. For each type, we employ a different mechanism. Under the aforementioned assumptions for parallel composition of differential privacy, the resulting values are ε-differentially private. Note that for each attribute, a different privacy parameter ε may be chosen. This way, the level of protection may be adapted to the sensitivity of the respective attribute values.

Numerical Attributes: For numerical attributes, like the age of a patient, we apply the bounded Laplace mechanism [36], an additive noise mechanism for numerical values. It draws noise from a Laplacian distribution, that is calibrated based on the privacy parameter ε and the sensitivity of the data distribution. The sensitivity Δf depends on the domain of the attribute d. We set the sensitivity for each attribute d to $\Delta f(d) = min(X_d(L)) - max(X_d(L))$, where the maximum and minimum are either user-defined or determined

by the values X_d of d in the original log L. To ensure only realistic values are drawn, we set bounds to the potential domain of attribute values based on the maximum and minimum value of d.

Categorical Attributes: For categorical attributes, such as the diagnosis of a patient, we apply the exponential mechanism [96]. It enables the definition of a utility difference between the different potential values of the domain of the categorical value. The probability of a value being exchanged by another value depends on the introduced probability loss.

Boolean Attributes: To ensure differential privacy of boolean data, one can use *randomized response* [170]. The algorithm is based on the following idea: A fair coin toss determines if the true value of an individual is revealed or if a randomized value is chosen instead. Here, the randomization depends on the strength ε of the differential privacy guarantee. We will use a so-called *binary mechanism* [63], a standard mechanism to handle binary attributes.

Anonymizing timestamps

To anonymize timestamps, we introduce random timestamp shifts, which are inspired by the treatment of network logs [181]. That is, we initially alter all timestamps based on some randomly drawn noise value, λ_{shift}, which is drawn, for instance, from a Laplacian distribution. The result is illustrated in the middle sequence of Fig. 7.2. After this initial shift, we subsequently introduce noise to the time intervals between events, depicted as Δ_1, Δ_2, and Δ_3 in the figure. To this end, we add random noise to the length of each interval, denoted by λ_1, λ_2, and λ_3. To retain the order of events, we bound the random timestamp shift to the size of the interval between two events. Since the event order was already anonymized in the first step of the framework (Section 7.2.1), introducing additional noise by re-ordering events here would just reduce the event log's utility.

After this final step, all aspects of the original log, i.e., control-flow and contextual information, have been anonymized. Based on the maxim of parallel composition, the resulting log provides ε-differential privacy.

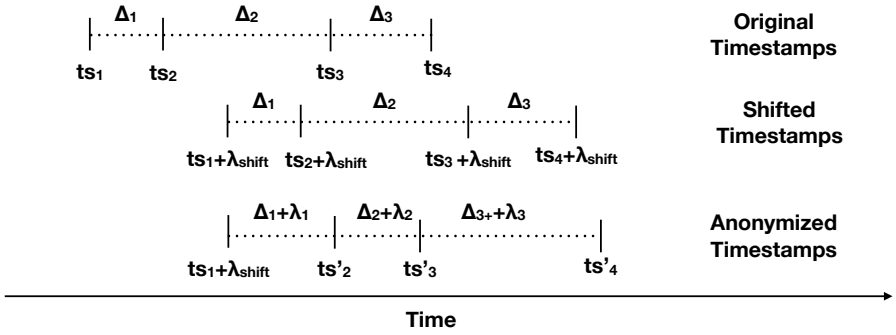


Figure 7.2: Illustration of timestamp anonymization.

7.3 Proof-of-Concept

This section presents a proof-of-concept of the PRIPEL framework. We first report on a prototypical implementation (Section 7.3.1), which we apply in a case study using a real-world event log (Sections 7.3.2–7.3.3). In this manner, we aim to show the feasibility of the framework in a realistic setting and investigate its ability to preserve the utility of an event log while providing privacy guarantees.

7.3.1 Prototypical Implementation

We implemented PRIPEL in Python and published it as a stand-alone tool under the MIT license on Github.* The implementation uses the PM4Py library [14] to parse and process event logs. For our proof-of-concept, we used the Laplace mechanism [92] as the trace-variant query approach. The anonymization of contextual information is based on IBM's *diffprivlib* library [62]. In addition to our stand-alone implementation, we also integrated PRIPEL into PM4Py.

7.3.2 Case Study Setup

We show the feasibility of PRIPEL by applying our implementation to the Sepsis event log [88]. We selected this event log given its widespread adoption as a basis for case studies, as well as due to the relevance of its characteristics in the context of our work. The reason being the long tail process behavior in terms of a relatively low number of re-occurring trace variants: 1,050 traces spread over 846 trace variants. As such, the log's challenging characteristics make it particularly suitable for a proof-of-concept with our framework.

*https://github.com/samadeusfp/PRIPEL

To parameterize our implementation, we test different values of the privacy parameter ε, ranging from 0.1 to 2.0. Given that this parameter defines the strictness of the desired privacy guarantees (lower being stricter), varying ε shall show its impact on utility of the resulting anonymized log. We select the maximal prefix length $l = 30$, to cover the length of over 95% of the traces in the Sepsis event log. To cover all potential prefixes of the original log, we would need to set $l = 185$. However, this would add a lot of noise and increase the runtime significantly. Therefore, we opt for only looking into shorter traces. For each event log, we opted for the lowest value for the pruning parameter p that still computes the query within a reasonable time, as will be detailed in the remainder.

7.3.3 Case Study Results

In this section, we first focus on the runtime of the PRIPEL framework. Subsequently, we explore its ability to preserve event log utility while guaranteeing ε-differential privacy.

Runtime

We measured the runtime of our PRIPEL implementation for various parameter configurations, obtained on a MacBook Pro (2018) with an i5 Intel Core CPU and 8GB memory. As shown in Table 7.1, we were typically able to obtain an anonymized event log in a manner of minutes, which we deem feasible in most application scenarios. However, the runtime varies considerably across the chosen configurations and the framework's three main steps.

Most of the anonymized logs have far more traces than the original log, due to the added noise as part of the trace-variant query. However, this is not true for the log with a $\varepsilon = 1.5$ differential privacy guarantee, which contains only one third of the number of traces of the original log. This is due to the low noise level and the fact that $p = 2$ cuts out all variants that appear only once. This applies to nearly all the variants in the original log. Since only a few noisy traces are added, the resulting log is significantly smaller than the original log. Furthermore, this log just represents one run of the algorithms. Another run of the algorithm could result in a log with a different size, since the Laplace algorithms produces high variance in terms of log size (see Chapter 6).

The trace-variant query (Step 1 in PRIPEL), is executed in a manner of seconds, ranging from one to nine seconds, depending on the configuration. However, this runtime could be greatly exceeded for configurations with a higher l. While a trace-variant query with $\varepsilon = 1.5$ and $p = 2$ is answered in one second, a configuration of $\varepsilon = 1.5$ and $p = 1$ does not lead to

ε	p	$\lvert\tau(L)\rvert$	Query	Enrichment	Anonymization	Total
0.1	20	5,175	1s	35s	3m24s	4m07s
0.5	4	6,683	1s	3m52s	4m08s	8m12s
1.0	2	7,002	2s	8m37s	4m27s	13m18s
1.5	2	340	1s	8s	13s	23s
2.0	1	13,152	9s	33m05s	8m30s	42m06s

Table 7.1: Runtime of *PRIPEL* for the Sepsis log

Attribute	Original	$\varepsilon = 2.0$	$\varepsilon = 1.5$	$\varepsilon = 1.0$	$\varepsilon = 0.5$	$\varepsilon = 0.1$
Infection Suspected (fraction)	0.81	0.75	0.69	0.67	0.58	0.51
Avg. Case Duration (days)	28.47	36.93	7.95	37.77	37.16	34.2
Median Case Duration (days)	5.34	11.23	0.12	11.92	10.95	9.57

Table 7.2: Sensitivity of attribute values to parameter ε

any result within an hour.

Sequence enrichment (Step 2) is the step with the largest runtime variance, from 35 seconds to 33 minutes. In most configurations, this step also represents the largest contribution to the total runtime. This is due to the polynomial runtime complexity of the enrichment step, see Section 7.2.2. To reduce the runtime, a greedy strategy may instead be used to match activity sequences and traces.

Anonymization based on local differential privacy (Step 3) has a reasonable runtime that increases linearly with the number of traces in the resulting log.

Based on these observations and the non-repetitive character of the anonymization task, we argue that it is feasible to apply our PRIPEL framework in real-world settings. However, if runtime plays a crucial factor in an application scenario, it should be clear that a suitable parameter configuration must be carefully selected.

Event log utility

To illustrate the efficacy of PRIPEL, we analyze the utility of anonymized event logs. In particular, we explore measures for three scopes: (1) the event level, in terms of *attribute value quality*, (2) the trace level, in terms of *case durations*, and (3) the log level, in terms of overall *process workload*.

Data attribute values: At the event level, we compare the value distribution of data attributes

in the anonymized logs to the original distribution. The Sepsis log primarily has attributes with boolean values. The quality of their value distributions is straightforward to quantify, i.e., by comparing the fraction of true values in an anonymized log L' to the fraction in L. To illustrate the impact of the differential privacy parameter ε on attribute value quality, we assess the value distribution for the boolean attribute *InfectionSuspected*. As depicted in Table 7.2, the truth value of this attribute is true for 81% of the cases in the original log.

The anonymized distribution is reasonably preserved for the highest ε value, i.e., the least strict privacy guarantee. There, the distribution has 75% true values. However, the accuracy of the distribution drops for stronger privacy guarantees, reaching almost full randomness for $\varepsilon = 0.1$. This illustrates that the quality of attribute values can be preserved for certain privacy levels, but that they may be impacted for stricter settings. Note that these results are obtained by anonymizing individual values so that the reduced quality for stronger privacy guarantees is inherently tied to the notion of differential privacy and is, therefore, independent of the specifics of the PRIPEL framework.

Case duration. Next, we investigate the accuracy of the case durations in the anonymized logs. Unlike the previously discussed quality of individual event attributes, the quality of case durations is influenced by all three steps of the framework. Therefore, when interpreting the results depicted in Table 7.2, it is important to consider that the maximal length of a trace is bound to 30 events in the anonymized logs (due to the selection of parameter n), whereas the original log contains traces with up to 370 events. However, we can still observe longer case durations in the anonymized logs due to the added noise. Additionally, in all scenarios, the average case duration is higher than the median case duration. This indicates that the log contains several outliers in terms of longer case durations. All anonymized logs reveal this insight. We conclude that *PRIPEL* preserves insights on the trace level, such as the duration of cases to some extent.

Process workload. Finally, at the log level, we consider the total workload of a process in terms of the number of cases that are active at any particular time. Given that anonymized event logs can have a considerably higher number of traces than the original log, we consider the progress of the relative number of active cases over time, as visualized in Fig. 7.3. The red dots denote the original event log, while blue triangles represent the anonymized event log with $\varepsilon = 1.0$.

The figure clearly shows that the general trend over time is sustained, but the anonymized log shows a consistently higher workload than the original log. Furthermore, the variance

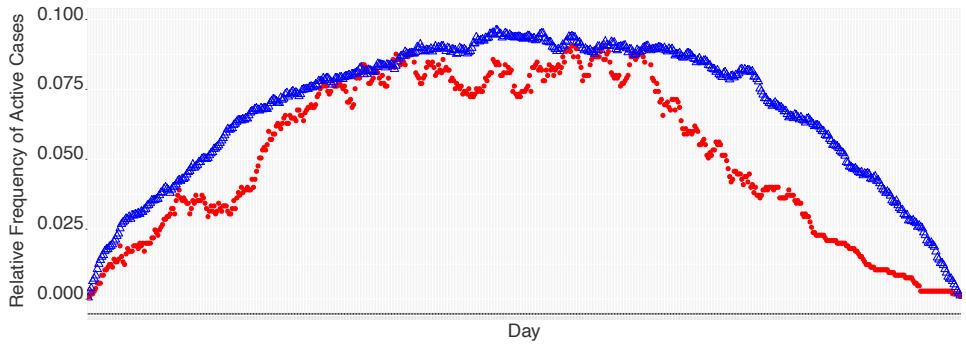


Figure 7.3: Active cases over time in original log (red) vs. anonymized log (blue).

over time is less extreme for the anonymized log. This shows that the noise insertion is reducing the variability. Nevertheless, the results illustrate PRIPEL's ability to preserve general utility for such a log-level process analysis.

7.4 Discussion

One Case per Individual

As already discussed above, the employed notion for differential privacy assumes that any individual, such as a patient, is only represented in one case. Alternatively, if the maximum number of cases per individual is known, the degree of noise introduced in the first step of the framework can be adjusted accordingly, by selecting the parameter ε. In general, multiple occurrences of individuals are also studied in the privacy literature [67]. If these multiple occurrences are limited to a small number of individuals the protection is still significant [67].

Attribute Dependency

For our approach, we assume that all attributes can be anonymized independently. Hence, the usefulness of anonymized values or the degree of privacy may be reduced for strongly correlated attributes. For instance, the independent anonymization of the *height* and *age* of a child may result in improbable combinations. Also, an attribute may represent a measurement that appears repeatedly in the trace, e.g., capturing the trend of a person's weight. Since the measurements are inter-related, the values to be anonymized are not independent, so that the parallel composition of differential privacy is not applicable. In that case, one can employ notions of differential privacy as developed for streaming settings [37].

Impact of Outliers on Sensitivity

The sensitivity in local differential privacy depends on the value of all possible values of an attribute. Hence, the sensitivity and the noise insertion can be strongly influenced by outlier values for attributes. Therefore, it might be necessary to reduce the number of outliers in an event log during pre-processing, in order to maintain the utility of the anonymized log.

Increased Duration of Traces

The timestamp anonymization technique limits how much a trace can be shortened, but provides no such bound for increasing the duration of the trace. Therefore, the traces generated by PRIPEL have a longer average duration than the traces in the original log. This limitation is of significance for certain time-based types of analysis and should be considered as a possible cause for error. An alternative timestamp technique could be developed and applied to address this issue.

Limits to Utility

Through PRIPEL a completely anonymized event log can be generated. In principle such an anonymized event log allows for any kind of analysis. However, it is important to note that local differential privacy induces a utility loss, i.e., in [65], it was shown that the utility loss of PRIPEL for process performance indicators is higher than the loss induced by algorithms tailored towards privacy-aware process performance indicators. Based on this observation, we recommend PRIPEL as a fall-back solution if the analysis performed with the anonymized data is unknown or multiple analyses shall be run on the same log. Nonetheless, in settings where a specific privatization technique is available, it is likely that such a tailored technique will provide higher utility than PRIPEL.

7.5 Conclusion

In this chapter, we introduced *PRIPEL*, a framework for publishing anonymized event logs that incorporates contextual information while maintaining differential privacy. Specifically, PRIPEL ensures local differential privacy by leveraging the maxim of parallel composition. Through a case study using a real-world event log of Sepsis cases from a hospital, we demonstrated that the utility of anonymized event logs is preserved for various types of analysis involving contextual information. We also highlighted that the PRIPEL framework and differ-

ential privacy in general are well-suited for analysis techniques that aim to aggregate or generalize over the traces in an anonymized event log, such as process discovery, log-level conformance checking, and process enhancement. However, it should be noted that the insertion of noise in the anonymization process may lead to the inclusion of data that never occurred in the original log, which could result in incorrect results when performing trace-level analysis such as establishing alignments for a single case.

This chapter is based research that was previously published in [54], in the Proceedings of the International Conference on Process Mining 2019 and as an article in Data & Knowledge Engineering [53].

8 PRETSA Algorithm Family

Event logs may contain sensitive information about service providers, such as employees, involved in the process execution. This chapter aims to address the risk of privacy-disclosure of service providers. To accomplish this, we introduce the PRETSA algorithms, a family of event log anonymization algorithms that provide group-based privacy guarantees in terms of k-anonymity and t-closeness. These algorithms protect the identities of service providers, prevent the identification of service provider membership in the event log, and prevent the disclosure sensitive attributes, such as performance information. They achieve this protection by merging traces that violate the privacy guarantee into similar traces. This strategy leads to a reduction in the process behavior covered by the control flow, as visualized in Fig. 8.1

The algorithms transform a prefix-tree representation of an event log, but differ in their handling of the trade-off between computational complexity and the utility of the anonymized event log for downstream analysis. We show by experiments that the PRETSA algorithms result in event logs with higher utility compared to baseline methods.

Within the remainder of this chapter, we first outline the problem we want to address in Section 8.1. Next, we formalize the attack and the anonymization problem we want to solve in Section 8.2. We then introduce the PRETSA algorithm family in Section 8.3. In Section 8.4, we provide an experimental evaluation of the algorithms. We provide a qualitative discussion of the PRETSA algorithms in Section 8.5. We conclude the chapter in Section 8.6.

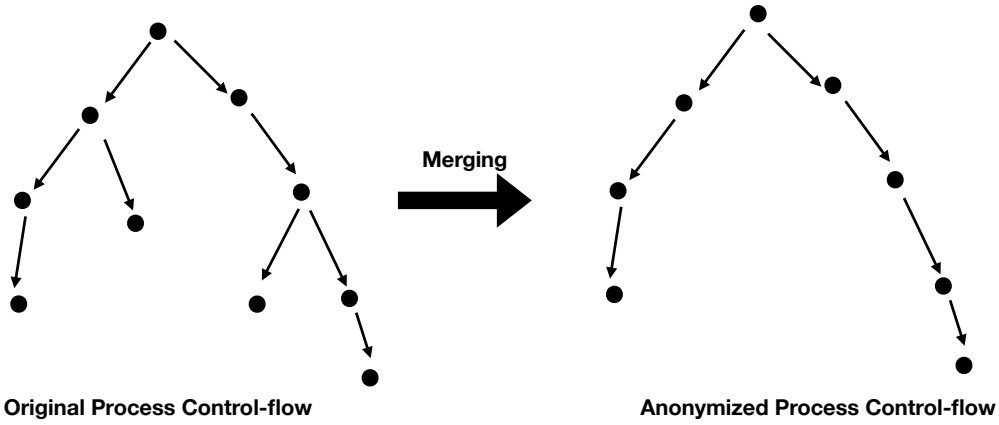


Figure 8.1: Process Control-flow visualized as a Prefix-Tree to provide an Intuition of Anonymization by Merging.

8.1 Problem Illustration

To motivate the problem of privacy protection for service providers, let us consider an internal purchase order (PO) process. It encompasses the following activities: the creation of a purchase order (*create_po*), updating them (*update_po*), receiving goods (*receive_gd*), and checking, paying, and rejecting invoices (*check_in*, *pay_in*, *reject_in*). Assume that events are recorded for this process in an event log, as exemplified in Table 8.1. Here, a total of 28 traces, i.e., sequences of activity executions as part of a specific process instance, have been recorded. As shown in Table 8.1, the traces can be grouped into five variants (v_1–v_5), depending on the sequence of activities that have been executed. Yet, as shown in Table 8.2, the durations of the activity executions differ among the traces, even when they are of the same variant.

Table 8.1: Exemplary event log that comprises sequences of activity executions.

	List of Trace Variants	#
v_1	*create_po*, *update_po*, *receive_gd*, *check_in*, *pay_in*	10
v_2	*create_po*, *update_po*, *receive_gd*, *check_in*, *reject_in*	5
v_3	*create_po*, *receive_gd*, *update_po*, *check_in*, *pay_in*	7
v_4	*create_po*, *receive_gd*, *update_po*, *check_in*, *reject_in*	5
v_5	*create_po*, *receive_gd*, *update_po*, *update_po*, *check_in*, *pay_in*	1

Table 8.2: Exemplary event log with durations of events.

	Traces incl. durations of activity executions
...	...
ξ_{21}	*create_po(s:1), update_po(s:10), receive_gd(s:1), check_in(s:5), reject_in(s:2)*
ξ_{22}	*create_po(s:1), update_po(s:8), receive_gd(s:2), check_in(s:2), reject_in(s:2)*
...	...
ξ_{31}	*create_po(s:1), receive_gd(s:1), update_po(s:7), check_in(s:25), pay_in(s:3)*
ξ_{32}	*create_po(s:1), receive_gd(s:3), update_po(s:11), check_in(s:4), pay_in(s:1)*
...	...

Privacy violations

Ethical and legal considerations prevent organizations from collecting or disclosing process-related data that compromise the identity of individual service providers. Hence, an event log must not contain information about which service provider performed which events. However, for someone with malicious intent, i.e., an *adversary*, information on the sequencing of events may be enough to relate service providers to the execution of certain events [6, 99]. This, particularly, holds if an adversary possesses organizational knowledge, since it might be possible to relate such knowledge to the traces in an event log. In this manner, an adversary might be able to derive sensitive information, such as:

- That an event was performed by a specific service provider (*identity disclosure*).
- That the data of a specific service provider is included in the event log (*membership disclosure*).
- That a service provider can be characterized by execution-related data, e.g. performance data (*attribute disclosure*).

For example, consider a scenario in which some POs have been updated after goods receipt. If an adversary knows that Sue is one of the few service providers who are allowed to subsequently check the corresponding invoice, the adversary would be able to identify the specific events that were performed by Sue (identity disclosure) with high accuracy. Similarly, the information that Jim is the service provider, who gets assigned high volume orders, for which checking the invoice takes a relatively long time, would enable the adversary to identify the respective events (attribute disclosure).

Privacy guarantee

To reduce the probability that such an attack will succeed, event logs must be anonymized. By utilizing *k-anonymity* we can bar the disclosure of infrequently occurring process behavior and at the same time, prevent the insertion of obfuscated control-flow behavior common to differential privacy. In the same vein, the ability to identify service providers based on data linked to the execution of an activity may be prevented by *t-closeness*, which limits the difference between value distributions observed for different equivalence classes.

Consider again the event log from Table 8.1. A straightforward manner to ensure k-anonymity would be to remove any variant from the log that occurs less than k times. Given that only one variant occurs at least eight times, a requirement for k-anonymity with $k = 8$ would yield a sanitized event log that contains only 10 traces that all represent variant v_1. To ensure *t-closeness*, in turn, long durations of all events denoting a check of an invoice may be removed. Adopting a threshold of 10, for instance, the sanitized event log would lack a duration value for event *check_in* as part of trace ξ_{31}.

Event log utility

From a process mining perspective, a downside of such guarantees is that information may become obscured by anonymization. When retaining only the traces that occur at least k times, a considerable amount of information on the presence and frequency of other sequence variants is hidden. In the aforementioned example, the anonymized log would contain information on only 1 out of the original 5 variants and on only 10 out of 28 traces. Moreover, removal of attribute values would perturb the statistical basis for process mining. When applying process discovery techniques on this anonymized log, we would discover a process model that only captures a fraction of the actually recorded process behavior and which may provide misleading results when evaluating the process' performance, e.g., the average cycle time.

To tackle this issue, we propose techniques to reduce the impact of anonymization on event logs while still providing the same privacy guarantees. The intuition underlying our work is that we recognize that it can be worthwhile to transform traces from certain uncommon ($n < k$) variants into other, similar variants. This way, we are able to preserve more trace variants, and thus event log utility, overall, while also guaranteeing behavioral correctness, i.e., all traces in the anonymized event log correspond to behavior that actually occurred in real life. For instance, in Table 8.1, one can recognize that variants v_2 and v_4 are highly

similar (only two events are swapped). By transforming the 5 traces of variant v_4 into ones of v_2 (or vice versa), we would obtain a k-anonymized event log that captures information on 20 (as opposed to only 10) out of 28 traces. The same mechanism also enables us to ensure t-closeness. That is, the duration of an event that is part of a transformation can be derived based on the values that are already present in the log for the respective activity, before the whole set of values is anonymized by noise insertion.

8.2 Privacy Model

In this section, we introduce a formal model of the trace linking attack (see Section 8.2.1), and the proposed privacy model to limit the success of the attack (see Section 8.2.2). Finally, we formally define the optimal event log anonymization problem (see Section 8.2.3).

8.2.1 Attack Model

We consider a scenario, in which an organization wants to analyze the performance of different trace-variants. One important principle of privacy is data minimization. Therefore, we assume only the minimal data for such an analysis is included within the event log L. In our case, that means that each event e has only one sensitive attribute $e.s$, an event identifier $e.i$, and an attribute to encode the service provider $e.r$. We define $\mathcal{S}$ as the universe of values for the sensitive attribute. Let $\mathcal{I}$ be the universe of all event identifiers and let $\mathcal{R}$ be the universe of service providers. We assume that the sensitive attribute is relevant for the process analysis.

Examples for sensitive attributes include the duration or cost associated with the execution of an event. To avoid privacy loss, all attributes that are not relevant for the analysis may simply be discarded. An event logs allows an adversary to gain gain information about individual service providers, i.e. to learn which service provider executed which activity. Therefore, we should not disclose the attribute service provider $e.r$ directly, to protect them from this information gain by the adversary.

Instead, we publish a projection of an event $e \in \mathcal{E}$. Previously, we defined $\mathcal{E}$ as the universe of events with each event denoting one activity $a \in \mathcal{A}$. Furthermore, declared that each event e can have a varying set of event attributes. We define our projection as $\pi : \mathcal{E} \to \mathcal{I} \times \mathcal{A} \times \mathcal{S}$ with $\pi(e) = (i, a, s)$ that removes information on the service provider from an event. This projection is lifted to a trace and a log, respectively, by applying it to all contained events, i.e., $\pi(\xi) = \langle \pi(e_1), \ldots, \pi(e_n) \rangle$ and $\pi(L) = \{\pi(\xi_1), \ldots, \pi(\xi_l)\}$.

As a consequence the projected log $\pi(L)$ prohibits direct conclusions on who performed which activity execution (i.e. which event) (*identity disclosure*), whether events that belong to a certain service provider are in the log (*membership disclosure*), or on a characterization of service providers by values of the sensitive attribute (*attribute disclosure*). However, it might be possible to reveal such information, by incorporating background information.

Background knowledge

We assume that an adversary may possess background knowledge on the relation between service providers and activities. In practical settings, such information is often available: It may stem, for example, from the definition of organizational responsibilities (different service providers shall execute different sets of activities) [18]; information extracted from shift schedules (service providers may differ in when they execute certain sets of activities) [185]; or role-based access control in information systems (different service providers can execute different sets of activities) [56].

We assume that the adversary is in possession of some powerful background knowledge that does not only provide insights into potential assignments of activities to service providers, but allows for limiting these assignments based on sequences of activities. Therefore, we can model application contexts that control the assignment of activities to service providers in a fine-granular way. In Section 8.1 we outlined this in the case of a invoice handling process. Let us assume that in this example only specific service providers can execute the payment of an invoice (*pay_in*), if the execution of the process is abnormal. Such abnormal behavior could be cases where the purchase order was updated (*update_po*) after the goods had already been received (*receive_gd*).

The background knowledge is formalized as a function $b : \mathcal{A}^* \times \mathcal{A} \to 2^{\mathcal{R}}$, so that $b(\langle a_1, \ldots, a_n \rangle, a) = \{r_1, \ldots, r_m\}$ captures that an activity a in a trace in which activities $a_1, \ldots, a_n$ have been executed already in the respective order, may be assigned to one of the service providers $\{r_1, \ldots, r_m\}$. For example, $b(\langle \textit{receive\_gd}, \textit{update\_po} \rangle, \textit{pay\_in}) = \{\textit{Per}, \textit{Sue}, \textit{Amy}, \textit{Jim}\}$ models that *Per*, *Sue*, *Amy*, and *Jim* may pay an invoice of a PO that was updated after goods receipt.

Trace linking attack

Based on this background knowledge, we consider a specific type of sequence linking attack [99] on a project event log. We define it as a *trace linking attack* on a given projected

log $\pi(L)$ attempting for the following privacy violations:

- *Identity disclosure*: To determine that a given event is performed by a specific employee, an adversary aims to identify a function $work : \mathcal{I} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{R}$, which assigns an event (i, a, s) of a projected trace $\xi \in \pi(L)$ to a resource r, such that (i, a, r, s) corresponds to an event of the original trace $\xi \in L$, i.e., $\xi' = \pi(\xi)$.
- *Membership disclosure*: To determine that the data of a specific service provider is included in an anonymized event log, an adversary aims to identify whether, for a service provider $r' \in \mathcal{R}$, there exists a projected trace $\pi(\xi) \in \pi(L)$, such that the original trace $\pi(\xi') \in L$, $\xi = \pi(\xi')$, contains an event (i, a, r, s) for some $i \in \mathcal{I}$, $a \in \mathcal{A}$, and $s \in \mathcal{S}$.
- *Attribute disclosure*: To determine that a service provider can be characterized based on execution-related data, we consider a given distance function over two bags of values of the sensitive attribute, $\mu_S : \mathcal{B}(\mathcal{S}) \times \mathcal{B}(\mathcal{S}) \rightarrow \mathbb{R}$, for an activity $a \in \mathcal{A}$. Then, an adversary aims to identify whether the distribution of values of events related to activity a differs by at least $\psi \in \mathbb{R}$ for some service provider $r^* \in \mathcal{R}$, i.e., $\mu_S(\sum_{r' \in \mathcal{R}} \Gamma(r', a), \Gamma(r^*, a)) > \psi$ with $\Gamma(r, a) = [s \mid \langle e_1, \ldots, e_n \rangle \in L, 1 \leq j \leq n : e_j = (i, a, r, s)]$.

Background knowledge is exploited for the above attack as follows. The background knowledge for a projected log $\pi(L)$ induces equivalence classes:

Each activity pattern of an element $(\langle a_1, \ldots, a_n \rangle, a)$ in the domain of b defines a class that contains a projected trace $\langle (i'_1, a'_1, s'_1), \ldots, (i'_m, a'_m, s'_m) \rangle \in \pi(L)$, if the trace shows the pattern, i.e., there exists a mapping $\lambda : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ such that $a_j = a'_{\lambda(j)}$ for $1 \leq j \leq n$ and $\lambda(j) < \lambda(j+1)$ for $1 \leq j < n$. In the worst case, each prefix of activities $\langle a'_1, \ldots, a'_m \rangle$ that exists in the projected $\pi(L)$ induces an equivalence class.

For each activity in each equivalence class, the set of service providers that could have been involved can be defined based on the background knowledge. In the above example under usage of the aforementioned background knowledge, we derive an equivalence class that contains a total of eight traces i.e. all traces of the variants v_3 and v_5. Through the background knowledge it is revealed that $\{Amy, Jim, Per, Sue\}$ could have executed the payment of the invoice.

A trace linking attack may disclose identity, membership, and/or attribute values for all of the equivalence classes of an event log. Let us consider the risk of identify disclosure for the example event log. We assume that at most four invoices could have been paid by one service provider. As a consequence at most four events of traces in an equivalence class can relate

to one service provider. The identity disclosure for one service provider, say *Jim*, is correctly assigning events of the projected log to *Jim*. We can calculate the maximum probability of being successful in this attack. The probability is bounded by the ratio of the occurrences of events related to an activity that may have been performed by a specific service provider (four) and the total number of traces belonging to the equivalence class (eight). Consequently, the maximal probability of a successful attack and, therefore, of correctly assigning the events for invoice payments to *Jim* is $4/8 = 0.5$.

8.2.2 Privacy Guarantees

The probability of a successful trace linking attack can be reduced, if the projected event log has certain characteristics that lower the probability of disclosure. Through *k-anonymity* [141] that can be achieved with regards to identity and membership disclosure. In our setting we define *k*-anonymity as follows:

Definition 3. *(k-anonymity) Let $\pi(L)$ be a projected event log. $\pi(L)$ satisfies k-anonymity, if and only if each equivalence class of $\pi(L)$ contains at least k traces.*

We discussed above that an equivalence class of $\pi(L)$ is induced by the background knowledge b in our model: Each activity pattern of an element of the domain of b defines one equivalence class. k-anonymity requires that each equivalence class contains at least k traces, consequently k-anonymity provides us a bound on the maximum probability for the success of the aforementioned disclosures.

First, let us consider identity disclosure. Let $max(a)$ be the maximal occurrence of events that correspond to the execution of an activity $a \in \mathcal{A}$ by one of the service providers. Then, the maximal probability of successful identity disclosure for an event e of a projected trace is bounded by $P(r \in work(e)) \leq max(a)/k$. In other words, guessing the correct assignment of a service provider to an event ($r \in work(e)$) in the equivalence class induced by the background information will succeed is bound to a probability of at most $max(a)/k$, as a consequence of k-anonymity.

In the aforementioned example, the previously discussed equivalence class contains eight traces. This equivalence class, therefore, yields 8-anonymity when only considering the invoice payment (*pay_in*) and yields a bound of $4/8 = 0.5$. If the projected log would satisfy 12-anonymity, we would get a lower bound for the probability of successful identity disclosure: The maximal probability of correctly assigning events related to the invoice payments to *Jim* would drop to $4/12 = 0.33$.

Furthermore, the notion of k-anonymity offers protection against membership disclosure. We consider a membership disclosure to be a success if it can be assumed with certainty that an event belonging to a certain service provider is part of the projected log. Nonetheless, this is not possible if each equivalence class has at least k traces, so that $k > max(a)$ holds true for all activities $a \in \mathcal{A}$. Because in such a setting at least two different service providers could have been responsible for the events of each equivalence class. As a consequence it is impossible to conclude with certainty that an event is associated with a specific service provider.

However, even if the projected log satisfies k-anonymity no protection against attribute disclosure can be ensured. To illustrate this lets consider that the values of the sensitive attributes of the events from one activity within an equivalence class may have a distribution that differs very much from the one over all events of the respective activity. Such a scenario would enable an adversary to derive conclusions linked to the service providers associated with the events in this equivalence class.

Protection against attributes disclosure attacks can be achieved through the adoption of an enhancement of k-anonymity, so called *t-closeness* [80], a privacy guarantee that is specifically tailored towards the protection of attribute values distributions. The notion of t-closeness limits the amount of information an adversary can gain through the sensitive attribute of domain $\mathcal{S}$, as it is quantified by a distance function $\mu_S : \mathcal{B}(\mathcal{S}) \times \mathcal{B}(\mathcal{S}) \to \mathbb{R}$ for the respective value distributions. Here, we consider two distance functions proposed in literature:

(1) The Earth Mover's Distance (EMD) [131] is proportional to the minimum amount of work needed to transform one distribution into another one, where a unit of work denotes a move of a weight of one by a distance of one.
(2) The stochastic distance function [32] measures the distance between two distribution by comparing the probability of a value failing within a certain interval of the distribution.

While both functions formulate a distance for value distributions, we will later see that the differences in their operalizations have implications for the algorithms to ensure t-closeness for an event log.

Given a specific distance function, we define t-closeness for our model, as follows:

Definition 4. *(t-closeness) Let $\pi(L)$ be a projected event log and μ_S be a distance function. An equivalence class of $\pi(L)$ shows t-closeness, if for all activities $a \in \mathcal{A}$, the difference in the value distributions over all events for a in $\pi(L)$ and in the equivalence class is bound by $t \in \mathbb{R}$, i.e., $\mu_S(\Omega(a), \Omega'(a)) \leq t$ with $\Omega(a) = [s \mid \langle e_1, \ldots, e_n \rangle \in \pi(L), 1 \leq j \leq n : e_j = (i, a, s)]$ being the distribution of the sensitive attributes for activity a and $\Omega'(a)$ as the restriction of $\Omega(a)$ to events of the equivalence class. $\pi(L)$ shows t-closeness, if all its equivalence classes show t-closeness.*

Through the privacy guarantee t-closeness we help to prevent attribute disclosure, by requiring that none of the equivalence classes induced by background information differs significantly, as defined by parameter t, in terms of the value distribution of the sensitive attribute. As such, it prevents any conclusion from the equivalence class on the service providers involved in the respective events through the values of the sensitive attribute.

8.2.3 Optimal Event Log Anonymization

We previously defined notions of k-anonymity and t-closeness as means to provide privacy guarantees for event logs. If a projected event log $\pi(L)$ fulfills both notions, the event log is guarded against trace linking attacks in accordance with the defined privacy guarantees. However, if $\pi(L)$ does not fulfill these notions, the event log needs to be transformed to fulfill the desired guarantees.

Definition 5 (Event Log Anonymization for Group-based Privacy Guarantees)**.** *Let L be an event log and k and t be parameters of the desired privacy guarantees. Then, event log anonymization is defined as a function $\zeta : \mathcal{L} \times \mathbb{N} \times \mathbb{R} \rightarrow \mathcal{L}$, such that the anonymized event log $\pi(L)'$ satisfies k-anonymity and t-closeness.*

As illustrated in Section 8.1, event log anonymization can considerably affect a log's contents, i.e., the traces, that are present in a anonymized, projected event log $\pi(L)'$. As such, the way in which anonymization is performed directly influences the utility of the resulting event log for process mining tasks. Therefore, event log anonymization shall aim to produce a anonymized event log $\pi(L)'$ that is as close as possible to the original event log $\pi(L)$.

We quantify the closeness of an anonymized log to the original one by a function that captures the cost that would be required to transform $\pi(L)$ into $\pi(L)'$. Let $\mu_\xi : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$ be a function that quantifies the distance between two traces. This distance may be purely syntactic and defined, e.g., as the string edit distance over the respective sequences of activity executions. However, more elaborated measures may also incorporate domain knowledge or rely on representation learning that incorporates the semantics of activities, such as the distance proposed in [128] based on the Act2Vec model [26]. In the remainder, however, we will assume that the distance is defined as the string edit distance when discussing the time complexity of the proposed algorithms. Let $map_\zeta : \mathcal{T} \rightarrow \mathcal{T}$ be a function, that returns the anonymized trace that was generated for the original trace. Given a trace distance measure,

we define a distance between projected event logs $\pi(L)$ and $\pi(L)'$, as follows:

$$\theta_L(\pi(L), \pi(L)') = \sum_{\xi \in \pi(L)} \mu_\xi(\xi, map_\zeta(\xi)). \tag{8.1}$$

Based thereon, we then define the problem of optimal event log anonymization as follows.

Definition 6 (Optimal Event Log Anonymization Problem)**.** *Let L be an event log, k and t parameters of the desired privacy guarantees, and μ_L be a log cost function. The event log anonymization problem is to derive an anonymized event log $\pi(L)' = \zeta(\pi(L), k, t)$, satisfying k-anonymity and t-closeness, such that the distance $\mu_L(\pi(L), \pi(L)')$ is minimal.*

We introduce algorithms that address this problem in the next section.

8.3 PRETSA-Algorithms Family

Having defined the problem of event log anonymization, this section introduces a family of algorithms to solve it or approximate a solution to it, respectively. We first give an overview of the algorithms (Section 8.3.1). Next, we introduce the basic *PRETSA* algorithm (Section 8.3.2). Subsequently, we introduce *PRETSA** (Section 8.3.3) and *BF-PRETSA* (Section 8.3.4), two search-based generalizations of the ideas of PRETSA.

Note that, to keep the notation concise, we henceforth assume all event logs to be projected. That is, we write L, L', L'' to refer also to the projected variants of the original event log and logs derived from it by some transformation.

8.3.1 Overview

We introduce algorithms to anonymize an event log, i.e., to transform an event log so that it satisfies k-anonymity and t-closeness. As defined in Section 8.2.3, this shall be done while preserving the utility of the event log. The latter requires us to keep the distance between the original log and the anonymized log small, ideally minimal. We build three algorithms to address this problem:

PRETSA anonymizes the event log by a greedy strategy. It models the event log as prefix tree. It merges prefixes that violate the privacy guarantee with the closest prefix. It is greedy, because PRETSA simply moves through the prefix tree to identify nodes that violate the privacy guarantees and handle those that are found.

PRETSA* formulates event log anonymization as a search problem. It instantiates a traditional A*-search to identify which transformations to apply in order to obtain the privacy guarantees with minimal loss in the utility.

BF-PRETSA is an adaptation of PRETSA* to cope with its exponential worst-case complexity. That is, BF-PRETSA adopts the formulation of a search problem, but implements a best-first-strategy, trading optimality (i.e., a minimal loss in data utility) for runtime performance.

We summarize the characteristics of the PRETSA algorithm and its derivatives in Table 8.3. While all of them guarantee k-anonymity, PRETSA and BF-PRETSA also provide guarantees for either variant of t-closeness, see Section 8.2.2. PRETSA*, in turn, is limited to the instantiation of t-closeness with the stochastic distance function. The reason is that employing EMD as a distance function potentially leads to non-determinism in the underlying A*-search. However, preventing such non-determinism, PRETSA* actually transforms the log such that the loss in data utility is minimal. Due to its best-first exploration of the search space, BF-PRETSA may not find this global optimum and, hence, derives an approximate solution to the problem defined in Section 8.2.3. By being guided through the search-based formulation of this construction, it yields a local optimization of data utility. As such, it avoids the high runtime of PRETSA* (see below for a detailed discussion of the time complexity), while achieving better utility than PRETSA, as we will later confirm empirically. Moreover, dropping the requirement to compute an optimal result, BF-PRETSA may also be used for both variants of t-closeness, whereas PRETSA* only supports the stochastic variant. Overall, BF-PRETSA thereby provides a compromise between runtime complexity and applicability, as well as optimality.

Table 8.3: Characteristics of the PRETSA algorithms.

Property	PRETSA	PRETSA*	BF-PRETSA
k-anonymity	√	√	√
t-closeness variants	EMD, stochastic	stochastic	EMD, stochastic
Data utility	ad-hoc	global optimum	local optimum
Complexity†	$\mathcal{O}(v^2)$	$\mathcal{O}\left(\binom{v}{2}^v\right)$	$\mathcal{O}(v^3)$

† v is the number of variants of the event log.

8.3.2 PRETSA

In this section, we introduce the *PRETSA*, an algorithm for *PREfix-Tree based event log SAnitization for t-closeness*, which provides a simple operationalization of the fundamental idea of our algorithm familiy. PRETSA is inspired by the BF-P2kA-algorithm (Brute Force Pattern-Preserving k-Anonymization) presented in [99] to sanitize personal data of sequential nature, such as sequences of visited locations. While PRETSA achieves event log anonymization in terms of the desired *k-anonymity* and *t-closeness* privacy guarantees, it realizes a greedy strategy. However, PRETSA does not yield an optimal solution: it only approximates a solution to the problem defined in Section 8.2.3. To illustrate this central point, we take up the traces of Table 8.1. The prefix-tree representation of the respective event log is shown in Fig. 8.2a. Setting $k = 6$, we detect several violations of k-anonymity. For instance, considering the path corresponding to trace variant ξ_2 of Table 8.1, i.e., *create_po*, *update_po*, *receive_gd*, *check_in*, *reject_in*, there may be background knowledge that induces an equivalence class that involves only the five traces of this variant, which violates k-anonymity for $k = 6$. PRETSA resolves this violation by merging the respective path with the one representing the closest trace variant, which is ξ_1, given as *create_po*, *update_po*, *receive_gd*, *check_in*, *pay_in*. Transforming all paths that denote violations, PRETSA generates the tree shown in Fig. 8.2b. While we discuss the example with a focus on k-anonymity, the guarantee for t-closeness is obtained in the same manner. Paths in the prefix-tree that comprise nodes that violate t-closeness are also resolved by merging the respective traces with those of their closest trace variant.

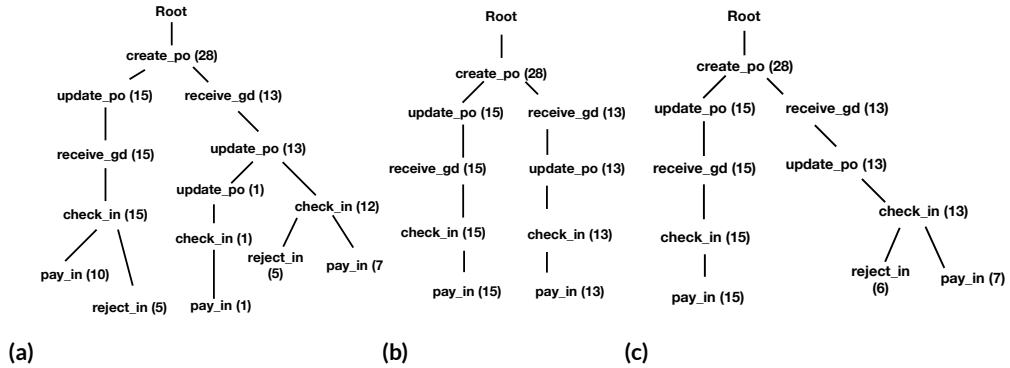


Figure 8.2: (a) Prefix-tree of the original example log; tree obtained for $k = 6$ with (b) PRETSA and (c) PRETSA*.

Let us look at the PRETSA algorithm in detail, as we outline it in Algorithm 5. First, the algorithm transforms the event log in a prefix tree (see line 1). Inside the prefix tree, each node

is a set of events that all represent executions of the same activity and share the same prefix of activity executions in their respective traces. We capture information about a node as a tuple $n = (a, T, S)$, where $a \in \mathcal{A}$ is the activity of the events; $T \subseteq \mathcal{T}$ is the set of traces that contain the events; $S \in \mathcal{B}(\mathcal{S})$ is the bag of values of the sensitive attribute. The algorithm traverses the prefix tree P in a depth-first manner (see line 4), until it reaches a node $n = (a, T, S)$ that violates the privacy guarantees (see line 5). If a violation is detected for node $n = (a, T, S)$, we disassociate the traces T from all of n's ancestors (see line 6). Next, we remove the respective node from the tree (see line 7). For each trace $t \in T$ of the pruned node n, PRETSA identifies a trace t' that is most similar (line 8). Here, we consider similarity in terms of the given trace distance function θ_ξ. Each trace $t \in T$ of the pruned node $n = (a, T, S)$ is then incorporated into all nodes $n' = (a', T', S')$, where T' contains the selected, most similar trace t', which involves adding the events once their activity a has been set to a' (line 9). For all events that have been transformed by PRETSA, the respective values of the sensitive attribute are discarded and replaced by a random value, which is drawn from the distribution of $\Omega(a)$ for events of activity a. The algorithm transforms the prefix tree iteratively, until it is fully traversed without identifying a single violation (line 11). Based on the obtained prefix tree, PRETSA returns a sanitized event log as the set of all traces represented by the tree (line 12). Next, we turn to the time complexity of PRETSA.

Algorithm 5: PRETSA

input : L, an event log; k and t, privacy parameters; θ_ξ, a trace distance function.
output : L', an anonymized event log.

```
1  P ← constructPrefixTree(L);
2  repeat
3      hasChanges ← false;
4      foreach n ∈ DFS(P) do
5          if violatesPrivacyGuarantees(t, k, n) then
6              updateAncestors(P, n);
7              prune(P, n);
8              t' ← findMostSimilar(P, n.traces, θ_ξ);
9              P ← reconstructTree(P, t')
10             hasChanges ← true;
11 until ¬ hasChanges;
12 return generateEventLog(P);
```

Theorem 1. *Given an event log with v trace variants, the time complexity of PRETSA is given as $\mathcal{O}(v^2)$.*

Proof. Considering Algorithm 5, we can observe that PRETSA will perform at most $v - 1$ merge operations, after we find at most $v - 1$ violations. However, for these merge operations it is necessary to compare the violating traces to all trace-variants, a calculation that can be done once upfront. The calculation of the distances between trace-variants is done v^2 times. Furthermore, after traces are removed from one node, the algorithm needs to check if the affected and already checked nodes still conform with the privacy guarantee. Such a re-checking can happen up to $v - 1$ times, but checking all affected nodes is equivalent to checking one additional variant. Checking the whole tree has a linear complexity, the additional checks necessary due to merges lead up to $v - 1$ checks. This in total results in $v - 1 + v$ checks, making the checking for privacy guarantee linear in v. Hence, we conclude that the calculation of the distances between trace-variants is the dominant factor and the time complexity for PRETSA is v^2. □

8.3.3 PRETSA*

The event log anonymization, as formulated in Section 8.2.3, can be phrased as a search problem. Here, event logs denote states in the search space and the transformation of one log into another one by merging traces, as realized by PRETSA based on the prefix-tree representation, induces transitions between these states. Final states are anonymized logs, i.e., those that satisfy the required privacy guarantees. The cost assigned to a state is determined by the distance of the respective log to the original log. As such, an optimal solution to the search problem is an anonymized log with minimal distance. Adopting this view, the anonymization may be approached by a search algorithm, as follows.

Instantiating A*-search for the above setting yields the PRETSA* algorithm, which is defined in Algorithm 6. It iteratively explores states of the search space and chooses those with the lowest predicted overall cost. To this end, it relies on the cost function f that assigns to a state (i.e., an event log) the cost of reaching the state from the initial state, captured by a function g, as well as the predicted cost from the current state to a final state, captured by a function h.

More specifically, the algorithm maintains a set of states to explore, *open* and a set of states that have been explored, *closed*. They are initialized with the original log and an empty set, respectively (line 1 - line 2) while the cost to reach the original event log, $g(L)$, is initialized with zero (line 3), so that the estimated overall cost, $f(L)$, is derived using the heuristic (line 4). The actual search is conducted as long as there are states to explore (line 5), selecting the state with the best estimated overall cost (line 6). It is added to the *closed* set (line 7). If the

respective event log satisfies k-anonymity, the function *noisify* adds noise to ensure t-closeness, as detailed below, so that the optimal anonymized event log is returned (line 8).

If the event log does not yet satisfy k-anonymity, the search considers all successor states, i.e., all event logs that can be obtained by merging the traces of two variants in the log (line 9). For each of these states, if they have not yet been explored (line 10), the cost of the path to the state is computed as a score (line 11). Also, the state is added to the *open* set, if it is not yet contained (line 12). If the state had been visited before, we check whether the currently explored path has a higher cost than the best known path (line 13) and if so, we continue with the next successor state. If not, the cost of the best known path to the current state and, based thereon, the estimated overall cost is updated (line 14 - line 15). Note that the algorithm returns an empty set if the log transformations (i.e., in each step merging all traces of two variants) do not yield a log satisfying k-anonymity (i.e., the number of traces is smaller than k).

The above algorithm includes two important design choices that we will discuss in detail below. First, we need to define the heuristic to guide the exploration of the search space. To do so, we define a function to assess the quality of an event log. Second, we instantiate the A*-search solely to achieve k-anonymity of an event log, but neglect violations of t-closeness. The reason being that A*-search requires the cost of a state to be deterministic, which cannot be guaranteed for transformations done to resolve violations of t-closeness. Therefore, to achieve t-closeness once k-anonymity is satisfied by an event log, we adopt noise insertion by function *noisify* in Algorithm 6. Later, we elaborate on the details of this function and we also discuss the time complexity of PRETSA*.

A Cost Model for A*-Search

The heuristic to estimate the cost to a final state, i.e., to an event log that satisfies k-anonymity, is based on the following intuition: Consider the traces of a trace variant. If these traces violate k-anonymity, i.e., then they will be merged with either the traces of the closest *non-violating* trace variant, or of the closest *violating* one. We therefore consider both options and incorporate the option with the lower cost.

To realize this idea, for an event log L', we distinguish the sets of traces of non-violating trace variants $V^+_{L'} \subseteq 2^{L'}$ and of violating trace variants $V^-_{L'} \subseteq 2^{L'}$. That is, an element $V \in V^+_{L'}$ is a set $V \subseteq L'$ that contains all traces of one variant and it holds $|V| \geq k$, i.e., k-anonymity is not violated. A set of traces $V \in V^-_{L'}$, in turn, represents a variant with $|V| < k$, i.e., a violation of k-anonymity.

Algorithm 6: The PRETSA* algorithm instantiating A*-search for event log anonymization.

input : L, an event log; k and t, privacy parameters; θ_ξ, a trace distance function; h, a heuristic to guide the search.
output: L', an anonymized event log.

$open \leftarrow \{L\}$; // Set of states to explore
$closed \leftarrow \emptyset$; // Set of explored states
$g(L) \leftarrow 0$; // Cost of best known path to state
$f(L) \leftarrow g(L) + h(L)$; // Cost of best known complete path through state
while $open \neq \emptyset$ **do** // While there are states to explore
  $L' \leftarrow \arg\min_{L'' \in open} f(L'')$; // Pick the best state according to heuristic
  $closed \leftarrow closed \cup \{L'\}$; // Record the state as explored
  if `is_k_anonymous`(L', k) **then return** `noisify`(L', t); // If k is satisfied, add noise
  for $L'' \in$ `derive_successor_states`(L') **do** // For each successor state
    if $L'' \in closed$ **then continue**;
    $s \leftarrow g(L') + \theta_\xi(L', L'')$; // Compute cost of new path to state L''
    if $L'' \notin open$ **then** $open \leftarrow open \cup \{L''\}$; // Add state to be explored
    else if $s \geq g(L'')$ **then continue**; // If new cost is worse, ignore path
    $g(L'') \leftarrow s$; // Set cost for best known path to state
    $f(L'') \leftarrow g(L'') + h(L'')$; // Set cost for best known complete path
return $\emptyset$;
function `noisify`(L', t) // Procedure to insert noise to achieve t-closeness
  $L \leftarrow \emptyset$; // Log to return
  for $\langle e_1, \ldots, e_n \rangle \in L'$ **do** // For each trace
    $\xi \leftarrow \langle\rangle$; // Noisy trace
    for $1 \leq j \leq n, e_j = (i, a, s)$ **do** // For each event
      $\xi \leftarrow \xi.\langle(i, a,$ `add_noise`$(s, t))\rangle$; // Add noise to sensitive attribute value
    $L \leftarrow L \cup \{\xi\}$; // Add noisy trace to log
  return L;

Using these auxiliary notions, we sum up the cost of merging all traces of violating trace variants into those of a closest non-violating variant (c_{cnv}) or a closest violating variant (c_{cvv}), which yields the predicted cost to a final state:

$$h(L') = \sum_{V \in V^-_{L'}} \min(c_{cnv}(V, L'), c_{cvv}(V, L')). \tag{8.2}$$

The heuristic for the cost of merging a variant into a closest non-violating variant, $c_{cnv}(V, L')$, is given by the respective distance for each trace that is merged. To operationalize this measure, we lift the trace distance θ_ξ, see Section 8.2.3, from traces to trace variants, i.e., $\theta_V : 2^{\mathcal{T}} \times 2^{\mathcal{T}} \to \mathbb{R}$ and for two trace variants $V, V' \subseteq L'$ the distance is defined as $\theta_V(V, V') = \theta_t(t, t')$

for some $t \in V$ and $t' \in V'$. Then, the cost of merging into a closest non-violating trace variant is:

$$c_{cnv}(V, L') = |V| \cdot \theta_V(V, V') \text{ with } V' = \underset{V'' \in V^+_{L'}}{\arg\min}\, \theta_V(V, V''). \tag{8.3}$$

Considering the cost of merging a variant into a closest violating variant, $c_{cvv}(V, L')$, we are not only relying on the respective trace distances. Rather, we also take into account that merging traces of two violating variants may resolve the violation of k-anonymity for both variants. Intuitively, our estimate here considers two situations: The minimal distance needed to merge the traces applies either $|V|$-times, when $|V| < k/2$, so that it is best to merge the traces into another variant; or $k - |V|$-times, when $|V| \geq k/2$, so that merging other traces into variant V is the best way to resolve the violation. In any case, the cost is no longer induced for both violating variants, but only one of them, so that only half of the cost is incorporated. Based on these arguments, the estimate is defined as follows:

$$c_{cvv}(V, L') = \frac{1}{2} \min(|V|, ||V| - k|)\theta_V(V, V') \text{ with } V' = \underset{V'' \in V^-_{L'}}{\arg\min}\, \theta_V(V, V'') \tag{8.4}$$

From the above definitions, it follows rather directly that the presented heuristic h is admissible, i.e., monotonic and never overestimating, as required by the A*-search algorithm. Both properties follow from the fact that in each step, the set of traces of violating trace variants, $V^-_{L'}$, is reduced, while for each set, a best-case estimate is incorporated. As such, the heuristic may underestimate the true cost. An example for that would be that the heuristic may calculate the cost of merging traces of a violating variant V based on a violating variant V', whereas the traces of V' are merged earlier into yet another one variant, so that the actual cost for merging V is higher. However, the heuristic never overestimates the cost.

Integrating t-closeness

The A*-search guarantees the construction of an optimal solution only under a deterministic cost function. While the resolution of violations of k-anonymity is based on a deterministic cost function, θ_V, the handling of t-closeness violates this assumption. The reason being that, following the approach introduced in PRETSA, we rely on the random generation of values of the sensitive attribute for artificially created events. Due to its stochastic nature, this

transformation cannot be incorporated into the A*-search directly.

Against this background, we propose to ensure t-closeness only after all violations of k-anonymity have been resolved, through function *noisify* in Algorithm 6. Here, we exploit the fact that stochastic t-closeness is closely linked to ε-differential privacy. As posed by Domingo-Ferrer et al. [32], a differentially private dataset fulfills stochastic t-closeness, so that we achieve the latter guarantee by noise insertion for the sensitive attribute. Specifically, the t-closeness guarantee provided by an ε-differentially private dataset depends not only on the privacy parameter ε, but also on the total number of records N of the dataset and the set of equivalences classes E into which they may be grouped [32]:

$$t = \max_{E} \frac{|E|}{N} \left(1 + \frac{N - |E| - 1}{|E|} exp(\varepsilon)\right). \tag{8.5}$$

In our setting, the equivalence classes to consider are the prefixes of traces in the event log after the violations of k-anonymity have been resolved using the A*-search. Then, to ensure t-closeness for a certain t, we must apply ε-differential privacy for the sensitive attribute for all events, while the privacy parameter ε depends on the number of equivalence classes. From the above equation, we derive the needed value for ε to be:

$$\varepsilon = \ln\left(\frac{(\frac{t \cdot N}{|E|} - 1)|E|}{N - |E| - 1}\right) \tag{8.6}$$

Incorporating this value for ε, we apply only the minimal level of noise insertion to the sensitive attribute of all events needed to guarantee t-closeness. However, the above relation between t-closeness and differential privacy holds only for the stochastic variant of t-closeness. It is therefore not applicable for the variant based on the Earth Mover's Distance when aiming at an optimal solution to the problem of event log anonymization.

Algorithmically, the above idea is formalized in function *noisify* in Algorithm 6. After an initialization of the event log to construct (line 18), we iterate over each trace (line 19) and construct a new, noisy trace (line 20) by adding each event (line 21) after noise has been added to the value of the sensitive attribute (line 22). The noisy trace is then added to the new event log (line 23), which is eventually returned (line 24).

Finally, we note that the above approach of ensuring t-closeness also has the advantage that it does not reduce the number of equivalence classes. This can be expected to be beneficial in terms of the utility of the anonymized event log.

Algorithm 7: The BF-PRETSA algorithm, which relaxes PRETSA*.

input : L, an event log; k and t, privacy parameters; θ_ξ, a trace distance function; h, a heuristic to guide the search.
output : L', an anonymized event log.

// $g(L), f(L)$ are initialized as in Algorithm 6
$L_{open} \leftarrow L$; // The currently explored state
while $L_{open} \neq \emptyset$ **do** // While the current state can be explored further
  if `is_k_anonymous`(L_{open}, k) **then return** `noisify`(L_{open}, t); // If k-anonymity is satisfied, add noise
  if `derive_successor_states`$(L_{open}) \neq \emptyset$ **then** // If the current state can be explored further
    $L_{best} \leftarrow \underset{L'' \in \texttt{derive\_successor\_states}(L')}{\arg\min} f(L'')$; // Pick the best successor state
    $g(L_{best}) \leftarrow g(L_{open}) + \theta_\xi(L_{open}, L_{best})$; // Set cost for best known path to state
    $f(L_{best}) \leftarrow g(L_{best}) + h(L_{best})$; // Set cost for best known complete path
    $L_{open} \leftarrow L_{best}$; // Consider the next state to explore
  else $L_{open} \leftarrow \emptyset$; // No further states to explore
return $\emptyset$;

Time Complexity of PRETSA*

PRETSA* constructs a solution to the optimal event log anonymization problem. However, satisfying the optimization problem is computationally expensive. Based on common complexity bounds for A*-search, we derive the following result on the time complexity of PRETSA*:

Theorem 2. *Given an event log with v trace variants, the time complexity of PRETSA* is given as* $\mathcal{O}\left(\binom{v}{2}^v\right)$.

Proof. In general, A*-search has a time complexity of $\mathcal{O}(b^d)$ with d as the depth, i.e., the length of the solution path, and b as the branching factor, i.e., the average number of successor states. The depth is bound by the number of trace variants, $d < v$, since PRETSA* will perform at most $v - 1$ merge operations for pairs of trace variants. Now, concerning the branching factor, we note that the number of possible successors of a state that represents an event log with k variants is $\binom{k}{2}$, i.e., the number of 2-element subsets of variants. The number of states with k variants, in turn, is the number of ways to partition the v variants into k non-empty subsets, i.e., the Stirling number of the second kind, $S(v, k)$, which is bound by

$1/2\binom{v}{k}k^{v-k}$ [125]. As such, the average branching factor can be derived by the total number of transformations $\sum_{k=2}^{v}\binom{v}{k}k^{v-k}\binom{k}{2}$ divided by the number of states, $\sum_{k=2}^{v}S(v,k)$. Here, a simple bound can be derived from the maximal branching factor, $\binom{v}{2}$. Adopting it as a bound, we have $b \leq \binom{v}{2}$. Moreover, we note that the check for k-anonymity requires a linear scan of the event log in the beginning to determine the number of traces per variant, and has a constant time complexity in the actual exploration of the search space. Also, the insertion of noise to ensure t-closeness is linear in the size of the event log and executed at most once. Hence, both functions do not add to the time complexity of the A*-search, which corresponds to $\mathcal{O}\left(\binom{v}{2}^{v}\right)$.

□

8.3.4 BF-PRETSA

PRETSA* anonymizes the event log so that the privacy guarantees, k-anonymity and stochastic t-closeness, are guaranteed, with minimal loss in the utility. However, as indicated by Theorem 2, the construction of an optimal solution has high computational costs.

Against this background, we also propose BF-PRETSA, which relaxes the A*-search employed by PRETSA* to achieve k-anonymity. That is, BF-PRETSA adopts the same definitions of the search space and the same heuristic cost functions as PRETSA*, but limits the expansions of search states to the best one available. As such, BF-PRETSA may not find the global optimum and provides solely an approximate solution when resolving violations of k-anonymity. However, due to adopting the formulation of event log anonymization as a search problem, BF-PRETSA enforces a local optimization and, hence, can be expected to yield better utility than the ad-hoc data transformation employed by PRETSA. By expanding only the best next candidate state (i.e., event log) in the search space, BF-PRETSA resolves violations of k-anonymity efficiently. To achieve t-closeness, BF-PRETSA realizes the same approach as PRETSA*, based on noise insertion for the sensitive attribute values of all events.

The above idea is formalized in Algorithm 7. Unlike PRETSA*, BF-PRETSA does not maintain a set of states to explore, but explores solely a single state in each iteration, referenced as L_{open} in Algorithm 7. Initially, this log is the original log (line 1). As long as a new state can be identified (line 2), we explore the search space. As before, an event log that satisfies k-anonymity is returned after noise has been inserted to ensure t-closeness (line 3). If that is not the case and if there exist successor states to explore (line 4), we select the best among these successor states (line 5). We then update the cost model (line 6 - line 7) and continue the exploration with the selected state (line 8).

Moreover, given that BF-PRETSA only strives for an approximate solution when ensuring k-anonymity, we can now employ both variants of t-closeness, i.e., based on the stochastic distance and on the Earth Mover's Distance.

The event logs obtained using BF-PRETSA are not guaranteed to provide an exact solution to the optimal event log anonymization problem. However, the approximate solution can be derived in polynomial time, as discussed next.

Theorem 3. *Given an event log with v trace variants, the time complexity of BF-PRETSA is given as $\mathcal{O}(v^3)$.*

Proof. Considering Algorithm 7, we first note that the main loop (line 2) of the algorithm is executed at most $v-1$ times, since at most $v-1$ merge operations for pairs of variants can be realized. In the loop, the check of k-anonymity can be done in constant time once the sizes of all trace variants have been determined upfront, in linear time in the size of the event log (as mentioned already in the proof for Theorem 2). Also, noise insertion to achieve t-closeness has a linear time complexity. However, in each iteration, we need to assess the quality of all successor states in order to pick the best one. There are $\binom{v-k+1}{2}$ such states in the kth-iteration. Taking the maximal value $\binom{v}{2} = 1/2(v-1)v$, observed in the first iteration, as a bound, we arrive at an overall time complexity of $\mathcal{O}(1/2(v-1)^2 v) \leq \mathcal{O}(v^3)$. □

We note that, with a cubic time complexity, BF-PRETSA is slightly less performant than the PRETSA algorithm, which has a quadratic time complexity. The difference stems from the fact that PRETSA merges a violating trace variant into a variant that shows a minimal string edit distance with the violating variant. Hence, all distances that are required by the algorithm can be computed upfront, for all pairs of variants. In BF-PRETSA, in turn, we adopt the cost model introduced for PRETSA*. Assuming that the same distance measure is used, however, we note that the cost is derived based on the sets of violating and non-violating trace variants, which potentially change in each iteration. As such, the distances cannot be computed upfront, but need to be determined in each iteration, which yields a cubic overall time complexity.

8.4 Evaluation

This section presents an experimental evaluation of the presented algorithms. By applying the algorithms on a collection of publicly available, real-world event logs, we assess how much

the algorithms need to change event logs in order to obtain desired privacy guarantees and, subsequently, assess how much event log utility is preserved.

In Section 8.4.1, we introduce the real-world event logs used in our experiments. The experimental setting for these experiments is described in Section 8.4.2, while Section 8.4.3 discusses the results.

8.4.1 Datasets

We use the same datasets for our evaluation as previously in Chapter 6. However, we preprocessed these logs by filtering out all variants that occur only once. This preprocessing step was necessary to ensure that PRETSA* terminates for at least some of the logs. In Table 8.4 we show the real-world event logs we used to conduct our experiments. We can see that the employed event logs differ considerably in various key aspects.

One important aspect is the number of traces per variant, which ranges from an average of 4.3 to 1138.4 over the different event logs. Given that this aspects influences the performance of the anonymization approaches under evaluation in a crucial matter, we believe that the utilized data collection is well-suited to achieve a high external validity of the results. The event logs also differ up to factor 10 in the number of variants. Since this aspect influences the runtime of our algorithms significantly, it enables us to examine the scalability of our solutions.

Table 8.4: Characteristics of the preprocessed event logs.

				Traces per variant	
Name	**Traces**	**Act.**	**Variants**	**Avg.**	**Max.**
Traffic fines [27]	150,270	11	132	1138.4	56,482
Hospital billing [94]	177,751	17	410	433.5	99,285
CoSeLoG [16]	1,348	16	30	44.9	713
BPIC 2013 [139]	1,236	6	76	16.3	485
Sepsis [88]	266	12	62	4.3	35

8.4.2 Experimental Setup

We used the following setup to conduct our evaluation experiments:

Algorithms

We compare the results obtained by the three different algorithms of the PRETSA family with each other. We showed in [54] that PRETSA outperforms simple baselines, we here focus on the comparison of PRETSA* and BF-PRETSA algorithms to PRETSA. Furthermore, we consider TLKC [117, 121], an anonymization technique is able to consider different kind of background knowledge. To ensure the best comparability with our algorithms, we run it using sequential background knowledge with a maximum length equal to the longest trace in the preprocessed log. Furthermore, we deactivate the attribute anonymization of TLKC and focus only on its control-flow anonymization. A previous study by Rafiei et al. [117] showed that TLKC provides superior results compared to naive anonymization strategies, such as simply removing violating variants. Therefore, we only consider TLKC as a baseline.

Parameters

We varied the strength of the desired privacy guarantees during the experiments, with $k \in \{4, 8, 16, 32, 64\}$ and $t \in [1, 5]$. For all event logs, we furthermore set the sensitive attribute S to the cycle time of an event, computed as the difference between an event's timestamp and the timestamp of its predecessor in the original log L.

Implementation and environment

We implement the algorithms in a stand-alone Python tool[*]. We use AnyTree[†] to implement the Prefix-tree.

All experiments were conducted on a Dell R920 server with an Intel Xeon E7-4880 CPU and 1 TB RAM. We used an execution timeout of 48 hours for each anonymization task, i.e., the application of an algorithm, with specific parameters, on a single event log.

Evaluation measures

We consider various measures to quantify the degree to which an event log was changed during the anonymization procedure:

Log distance: This measure captures the edit distance between the original log L and an anonymized log L'. It is computed as the sum of the edit distance between each trace

[*] https://github.com/samadeusfp/PRETSA
[†] https://anytree.readthedocs.io/en/latest/

$t \in L$ and its counterpart in the anonymized log, $t' \in L'$. Specifically, we employ the standard edit distance where each operation (removal or insertion) has equal cost.

Modified traces: We also consider the number of traces that were altered during the anonymization procedure, i.e., the traces for which at least one event was changed, added or deleted by the anonymization algorithm. Here, we only consider changes on the level of the activities referenced by the events, not the sensitive-attribute values.

Retained variants: Finally, we consider the number of variants that remain in L' after applying anonymization.

Additionally, we also quantify the impact of the anonymization procedure on the utility of an anonymized log L'. We achieve this through two measures, one to assess the control-flow utility and one related to the attribute-value utility.

DF-representativeness: To compute the utility from a control-flow perspective, we determine the representativeness of the directly-follows (DF) relation (capturing for which activities there exist events in traces that follow each other directly) for the anonymized log L' compared to the original log L. Specifically, we employ the measure proposed by Knols et al. [72], which considers both the completeness of the relation, as well as the relative frequency with which certain behavior occurs.

Mean attribute-value error: To quantify the utility of the anonymized attribute values, we also compute the error introduced to the sensitive attribute S in an anonymized log L', which in our experiments corresponds to the mean cycle time of an activity. We use the relative error and take the average over all activities referenced in a log. Additionally, we normalize this metric to the range $[0, 1]$. For activities that are no longer present in L' due to anonymization, we assign the maximum error of 1.0.

8.4.3 Results

This section presents the results of our experiments. We first provide insights into the runtime of the algorithms on the real-world event logs, since this is an important distinguishing factor among the algorithms. Then, we provide an in-depth analysis of the results obtained using all three algorithms for the CoSeLoG event log, which is the only case where PRETSA* terminates within 48 hours. Afterwards, we consider the performance of BF-PRETSA for all considered event logs and show how it outperforms the PRETSA algorithm.

Runtime analysis

Fig. 8.3 shows the runtime of the three algorithms. We observe that BF-PRETSA has a higher runtime than PRETSA in all experiments. However, BF-PRETSA finished for all settings in less than 24 hours and is therefore capable of handling real-life event logs. In contrast, PRETSA* only terminates for one event log within 48 hours. Therefore, it is clear that the runtime of PRETSA* hinders its application to real-life event logs.

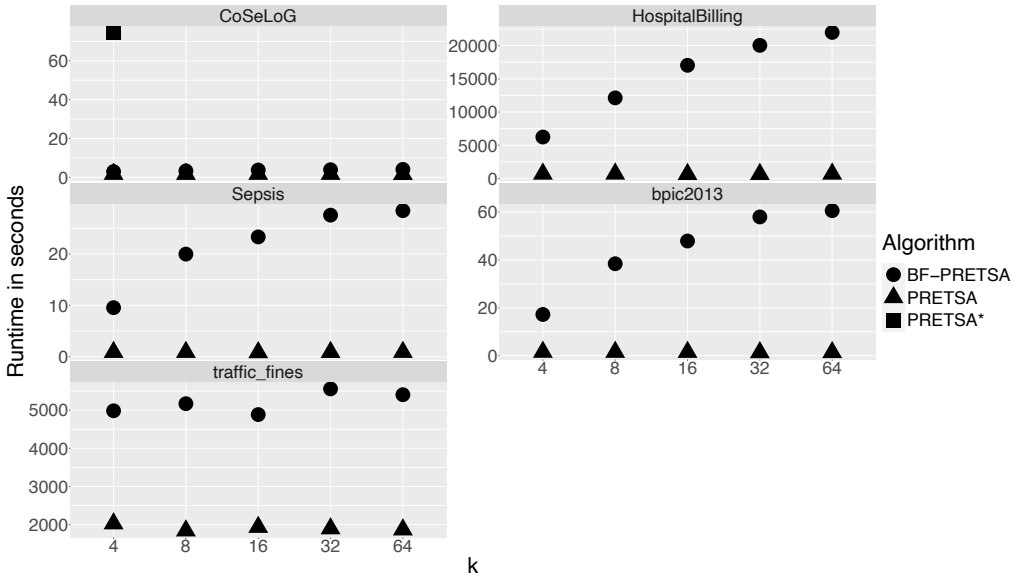


Figure 8.3: Runtime comparison.

Comparing PRETSA, BF-PRETSA, and PRETSA*

As shown in Section 8.4.3, due to its complexity, we were only able to obtain results using PRETSA* for CoSeLoG with $k = 4$ (while the value for t is varied). Still, the results obtained for this scenario can provide us with insights into how close the heuristic-based BF-PRETSA approach comes to the optimal results obtained using PRETSA*.

Table 8.5 provides an overview of the main results obtained for these settings. The table reveals that BF-PRETSA and PRETSA* outperform the PRETSA. At the same time, the difference between BF-PRETSA and PRETSA* is marginal, highlighting that BF-PRETSA approximates the optimal solution well. These trends hold across all considered measures as well as for all values of parameter t, as next investigated in detail.

PRETSA* vs. PRETSA

We observe that the anonymized event log L' much more closely reflects the original log L after applying PRETSA* in comparison to PRETSA. For example, for $t = 2$, PRETSA modified 379 traces out of the 1,348 total traces (28.1%), whereas the optimal solution from PRETSA* only required a modification of 12 traces (0.9%). Furthermore, out of 30 variants in L, the optimal solution retained almost twice as many variants than PRETSA, i.e., 24 versus 14 (or 80.0% versus 46.7%). Similarly, the utility of the anonymized log is much higher after applying PRETSA*. For instance, PRETSA* achieves a considerably higher DF-representativeness than PRETSA* (0.91 vs. at most 0.66), which shows that the behavior of the anonymized log L' (i.e., the control-flow utility) much closer reflects the behavior in the original log L after applying PRETSA*. When considering the utility of the anonymized attribute values (the activity durations), we also observe that PRETSA* introduces a considerably lower error than PRETSA, e.g., 0.08 versus 0.26 for $t = 5$.

Overall, we observe that BF-PRETSA often provides similarly good results as PRETSA*. One reason for this, is that both techniques only differ in their handling of k-anonymity but not their handling of t-closeness. We observed in our experiments, that our indirect way of guaranteeing t-closeness is beneficial over the original approach of removing violating traces. Therefore, we can attribute a huge part of the improvement of BF-PRETSA and PRETSA* over PRETSA to this novel handling of sensitive attributes.

PRETSA* vs. BF-PRETSA

When comparing the optimal results obtained using PRETSA* against the results of the heuristic-based algorithm, BF-PRETSA, we observe that the latter closely approximates the optimal results, especially when contrasted with the results obtained with PRETSA. For example, when considering the degree by which L' was changed, the number of modified traces (12 vs. 14) and retained variants (24 vs. 23) are comparable, whereas the control-flow utility quantified through the DF-representativeness measure is equal between the two. When considering utility in terms of the attribute error, we observe that BF-PRETSA sometimes achieves better results than PRETSA*, e.g., for $t = 2$, the error of BF-PRETSA is 0.08, whereas for PRETSA* it is 0.12, while for, e.g., $t = 4$, PRETSA* achieves a lower error. However, these fluctuations stem from the non-deterministic manner in which noise is inserted to ensure t-closeness. Therefore, they should not be interpreted as an indicator of the superiority of either algorithm.

Table 8.5: Results obtained for CoSeLoG with $k = 4$

Measure	Algorithm	t=1	t=2	t=3	t=4	t=5
Log distance	PRETSA	6,342	741	758	636	442
	BF-PRETSA	32	30	30	30	30
	PRETSA*	26	26	26	26	26
Modified traces	PRETSA	1,232	379	378	291	198
	BF-PRETSA	14	14	14	14	14
	PRETSA*	12	12	12	12	12
Retained Variants	PRETSA	1	14	14	15	12
	BF-PRETSA	23	23	23	23	23
	PRETSA*	24	24	24	24	24
DF-represent.	PRETSA	0.0	0.56	0.56	0.63	0.66
	BF-PRETSA	0.91	0.91	0.91	0.91	0.91
	PRETSA*	0.91	0.91	0.91	0.91	0.91
Attribute error	PRETSA	1.00	0.32	0.32	0.29	0.26
	BF-PRETSA	0.10	0.08	0.13	0.11	0.16
	PRETSA*	0.13	0.12	0.09	0.09	0.08

PRETSA vs. BF-PRETSA

Next, we compare PRETSA and BF-PRETSA over all event logs with varying strengths for the privacy parameters k and t. Our results show that PRETSA is continuously outperformed by BF-PRETSA, to sometimes extreme degrees. As an example, consider the number of modified traces in Fig. 8.4. The figure captures the ratio of the traces that are modified by either anonymization algorithm. This ratio is always smaller than one, indicating that BF-PRETSA modifies less traces than the existing PRETSA algorithm. In fact, in several scenarios, BF-PRETSA modifies only a quarter of the traces in comparison to PRETSA.

As shown in Fig. 8.5, the modifications of BF-PRETSA are also less costly, as captured by the ratio of the edit distances of the logs anonymized with either technique and the original log. Similarly, Fig. 8.6 shows that BF-PRETSA preserves more variants than PRETSA, in all but one scenario. In this exceptional case, PRETSA preserves three variants, whereas BF-PRETSA preserves two, so that the difference is small in absolute terms. Therefore, we conclude that, in general, BF-PRETSA outperforms PRETSA with regards to the introduced modifications to the log.

More modifications of an event log can be expected to yield a higher loss of utility. This

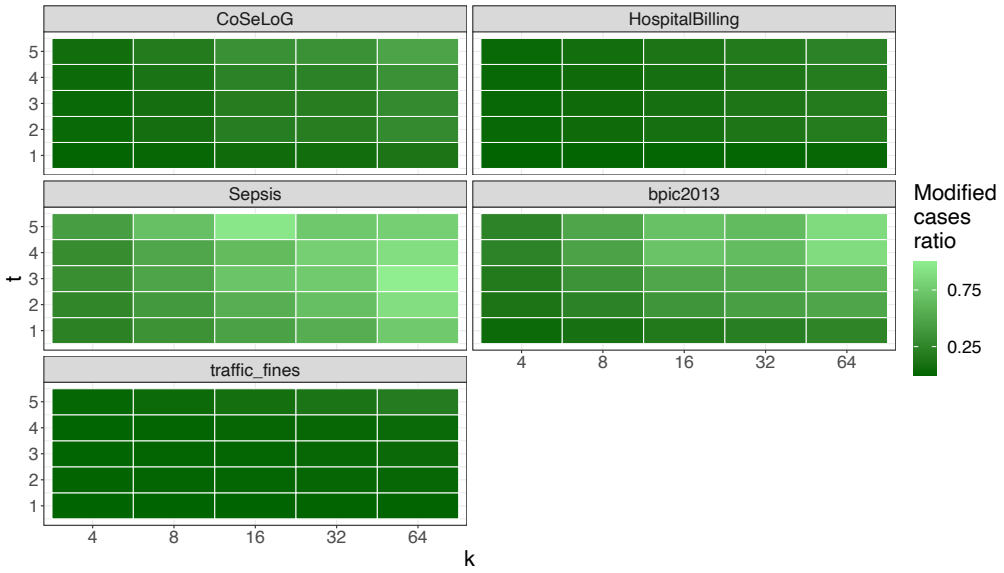


Figure 8.4: Ratio of modified traces, PRETSA vs BF-PRETSA.

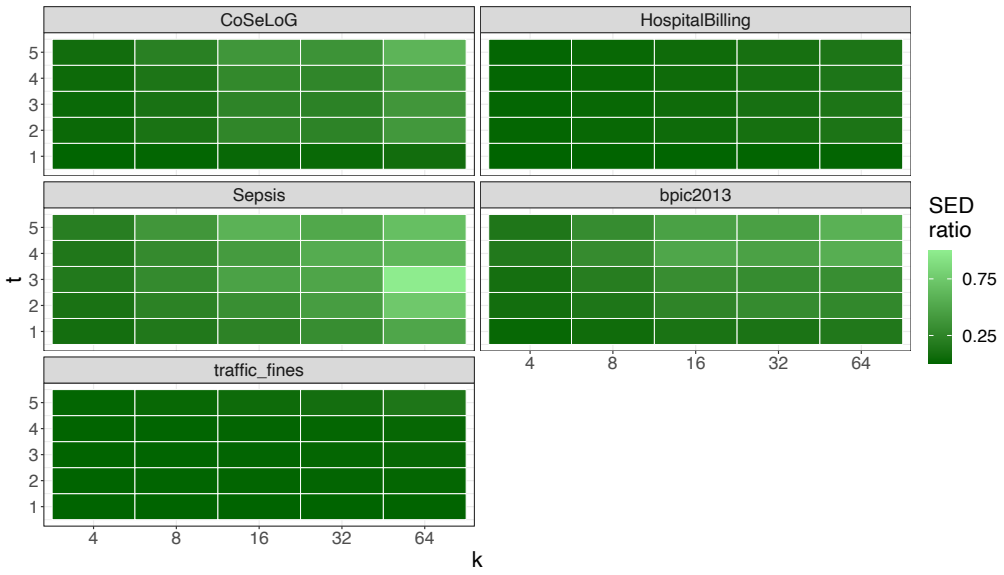


Figure 8.5: Ratio of edit distance, PRETSA vs BF-PRETSA.

assumption is supported by Fig. 8.7, which visualizes the increase in percentage points of the directly-follows representativeness obtained BF-PRETSA in comparison with PRETSA.

Figure 8.6: Ratio of retained variants, PRETSA vs BF-PRETSA.

Again, BF-PRETSA always outperforms PRETSA, while we see the highest improvement for low values for k. This is expected, since a low value of k induces a larger solution space, so that there is a larger potential to outperform a greedy algorithm.

It is also interesting to consider these observations in terms of modifications and utility loss in light of the characteristics of the different event logs. Particularly, we observe that BF-PRETSA achieves greater gains over PRETSA for smaller event logs, such as CoSeLoG, Sepsis, and BPIC 2013, whereas the performance of BF-PRETSA is comparable to that of PRETSA. In smaller logs, it is more likely that a larger fraction of variants occurs less than k times (given that k is set independent of the log size), which means that the benefits of more sophisticated event log anonymization, as achieved by BF-PRETSA, are more pronounced.

Similar observations are made for the mean attribute-value error. Fig. 8.8 illustrates the reduction to the introduced mean cycle time error in percentage points. Compared to PRETSA, BF-PRETSA considerably reduces this error across all evaluation scenarios.

Comparing BF-PRETSA vs. TLKC

We turn to compare BF-PRETSA with the TLKC approach. In Fig. 8.9, we show the number of remaining variants for both techniques for varying values of k. Notably, for three out of five

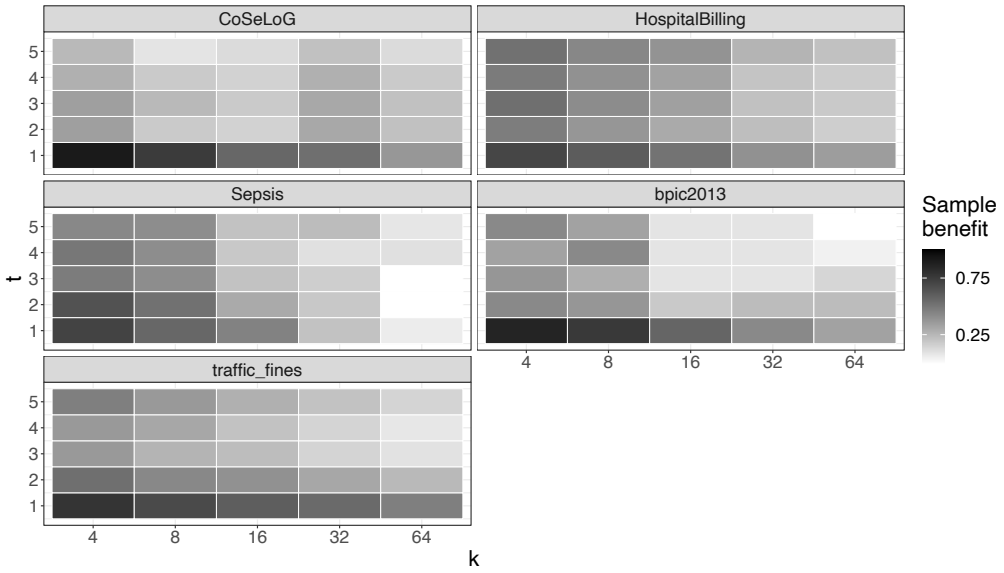


Figure 8.7: Increase of directly-follows representativeness in percent points, PRETSA vs BF-PRETSA.

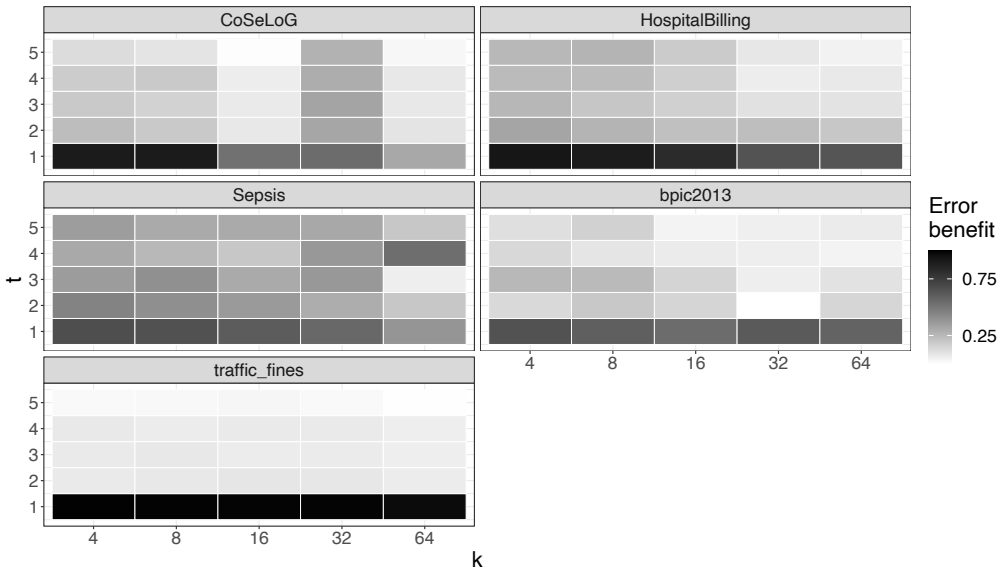


Figure 8.8: Reduction of mean relative error of cycle time in percent points, PRETSA vs BF-PRETSA.

event logs BF-PRETSA preserves more variants (Traffic Fines, Hospital Billing, BPIC 2013). This is especially true, for the large event logs Traffic Fines and Hospital Billing. In the case of the Sepsis event log, the most unstructured event log, TLKC is able to preserve more variants

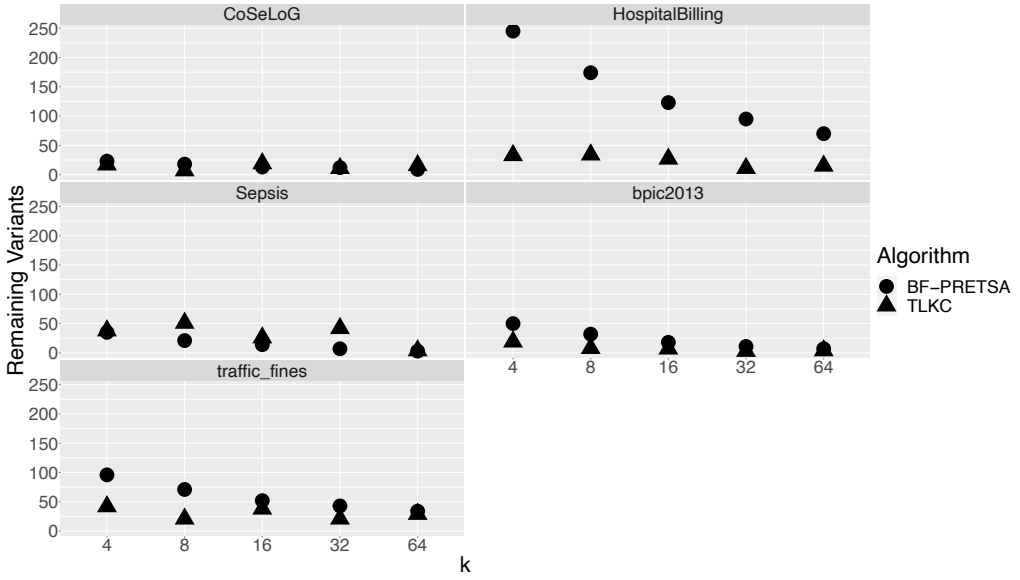


Figure 8.9: Number of retained variants, TLKC (dotted line) vs BF-PRETSA (straight line).

for some setting for k. However, it is important to note that TLKC can contain variants that have not been part of the original log, i.e. for one CoSeLoG setting, TLKC removes the most common start activity from all traces. Therefore, variants obtained with BF-PRETSA that represent prefixes from the original log provide an additional value. Since, an analyst can be sure that they appeared within the real process.

Next, we turn to the ratio of log distance that is calculated by dividing the log distance introduced by TLKC by the log distance introduced by BF-PRETSA. Consequently, values higher than 1 show a benefit for BF-PRETSA. We can observe such a benefit for all but one setting, as shown in Table 8.6.

Finally, we compare both techniques in terms of their ability to preserve directly-follows-relations. We show the results in Fig. 8.10. Again, BF-PRETSA outperforms TLKC on three out of five event logs significantly. For the other two events logs, we obtain mixed results. Overall, we can conclude that BF-PRETSA can provide significant higher utility that TLKC and seems to never underperform significantly compared to TLKC.

Table 8.6: Results obtained for Log-Distance-Ratio for TLKC vs. BF-PRETSA

Event Log	k=4	k=8	k=16	k=32	k=64
CoSeLoG	77.2	44.2	7	6.1	0.5
BPIC 2013	18.3	11.2	5.8	4.9	3.3
Hospital Billing	1017	518	281	173	121
Sepsis	2.6	1.1	1.1	1.7	1.4
Traffic Fines	102.5	356.2	23	108	10.7

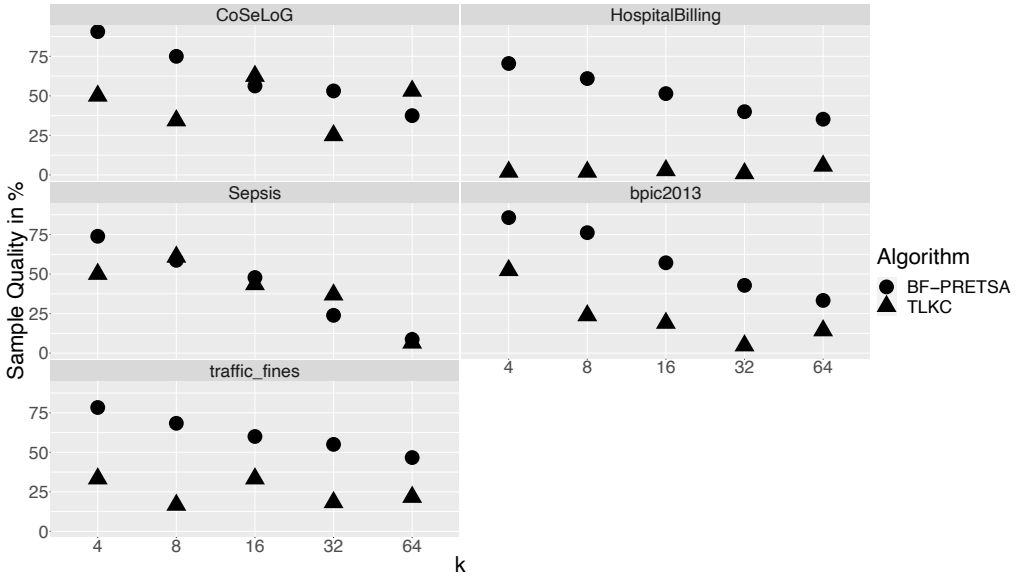


Figure 8.10: Comparison of directly-follows representativeness in percentage points, TLKC vs. BF-PRETSA.

8.5 Discussion

We now turn into a discussion about different characteristics and observations of the PRETSA-Algorithm family. Within Section 8.5.1, we discuss aspect that relate to the whole family. In Section 8.5.5, we discuss aspects specific to certain algorithms within the family.

8.5.1 Discussion related to the whole Algorithm Family

Privacy Guarantee

For group-based privacy guarantees, it is common to assume that certain parts of the published data can be exploited by the adversary as a quasi-identifier, but do not reveal sensitive information. In our setting, we defined the sequence of activities as quasi-identifiers. Noticeably, activities can encode sensitive information, i.e., the treatment of a certain disease. However, such information is often more sensitive for the service consumer than the service provider. Therefore, it is important to note that the PRETSA algorithm family is not the right tool to offer protection to service consumers. At the same time, if an activity would reveal sensitive information about a service provider, i.e., their current location in a mobile work setting, the PRETSA algorithm family is also not able to protect such information with a formal privacy notion.

8.5.2 Dependency on Distance Function

Within this chapter, we focused on generating anonymized logs that are close to the original log. We defined closeness using a distance function. In our case, and also common in the literature, the string-edit distance is used as a distance function for traces of an event log. The utility of the anonymized log will depend on how suitable the distance function is for the respective process mining analysis. Therefore, it is recommended to select a distance function based on the analysis that will be performed. However, so far, this topic has only been briefly examined in the scientific literature [128].

8.5.3 Replacing Prefixes with Suffixes

Most process mining techniques are performed on the prefix of the trace, with events being in ascending order of timestamps. However, some conformance checking techniques are performed on the suffix (i.e., the descending ordering of timestamps) **??**. At first glance, it could be argued that the PRETSA algorithm family could also be applied to a suffix tree. However, it is important to note that the background knowledge of the adversary is assumed to consist of prefixes. As a consequence, the anonymization of a suffix tree could lead to privacy violations based on the assumed attack model.

8.5.4 Utility for Process Discovery

Within this chapter, we did not include the results of a quantitative analysis of the algorithm with regards to process discovery. This was an intentional decision, after we performed respective experiments. The algorithms by design can not introduce new behavior, but only preserve existing behavior. This makes them hard to evaluate, since the models mined by them will produce high f-score values, even if they are not really useful for an analyst. This is due to the fact that BF-PRETSA and PRETSA* will in the worst case produce a log that only consists of the most common trace-variant. The same result is very likely for PRETSA. However, it is not guaranteed, because the most common trace-variant might be removed if it violates k-anonymity or t-closeness. A process model generated based on the most common trace-variant, will result in a perfect precision value, since all its control-flow will be represented within the original log. At the same time, we observed that models that just consist out of the most common trace-variant still get high values in terms of fitness. Consequently, we could not observe significant differences between PRETSA and BF-PRETSA within experiments that aim to measure the process discovery utility of the logs produced by these algorithms. Often, both algorithms produced models with the same utility.

8.5.5 Discussion for specific Algorithms

Runtime of PRETSA*

The runtime of PRETSA* limits is applicability for real-world event logs. If the heuristic employed within the A*-search is perfect, the algorithm would effectively have the same runtime as BF-PRETSA and at the same time always produce the optimal anonymized log. The reason is that PRETSA* would not explore a lot of different states, but instead mostly focus on one path towards a final anonymized log. Therefore, PRETSA* could become a viable option.

Multiple Sensitive Attributes with PRETSA* and BF-PRETSA

PRETSA* and BF-PRETSA are theoretically able to anonymize several sensitive attributes. The anonymization of sensitive attributes is based on noise insertion, and this step can be done independently for multiple attributes. Furthermore, it would be possible to use differential privacy mechanisms tailored towards preserving certain aspects for better utility. However, it is not entirely clear how multiple sensitive attributes would impact the privacy

guarantee, since they could also serve as quasi-identifiers and, therefore, weaken the privacy protection.

8.6 Conclusion

In this chapter, we focused on the problem of event log anonymization to protect service providers. We introduced the PRETSA algorithm family that is based on the idea of modeling an event log as a prefix tree. We started with the PRETSA algorithm, a greedy algorithm to ensure k-anonymity and t-closeness for such prefix trees by merging close variants. We generalized the underlying idea and presented *PRETSA** as an algorithm that is based on a search-based formulation of prefix-based event log anonymization. As such, it derives a solution that guarantees k-anonymity and t-closeness, while minimizing the utility loss. In the light of its high worst-case time complexity, we further presented the BF-PRETSA algorithm. It is based on the same search-based problem formulation, but adopts a best-first strategy that yields low runtime and close-to-optimal data utility. Our evaluation with five real-world event logs and several utility metrics illustrates that both PRETSA* and BF-PRETSA drastically improve the utility of the anonymized event logs compared to PRETSA, with BF-PRETSA also showing an acceptable runtime performance. Furthermore, we showed that BF-PRETSA outperforms TLKC in several application scenarios.

Part IV:

Final Remarks

9 Conclusion

With this chapter, we will end this thesis. First, we provide a summary of the results of this thesis in Section 9.1. We will also summarize the limitations of the presented approaches in Section 9.2. Finally, in Section 9.3, we will outline potential directions for future work within the research field of anonymization in process mining.

9.1 Summary of the results

As a first step, we provided a qualitative analysis of privacy threats within process mining. Based on these threats, we created a list of requirements for privacy-preserving process mining. We also provided a qualitative analysis of the re-identification risk in event logs, one of the major privacy threats we identified. We showed how anonymization can be used to address these requirements and to mitigate the identified threats. Therefore, we successfully motivated the need for anonymization techniques for event logs.

In the next step, we provided two control-flow anonymization techniques that guarantee differential privacy. Moreover, we showed how anonymized control-flows data can be enriched with contextual information. These techniques achieve their privacy guarantee through noise insertion, potentially inserting new behavior into the anonymized event log. However, these techniques aim to preserve the semantics of the control-flow and have shown superior performance compared to the state of the art.

Furthermore, we introduced a family of algorithms that aim to the syntactical preservation

of the anonymized event logs. The optimized algorithms within this family use a heuristic-based search to achieve this goal. These algorithms provide privacy protection through k-anonymity and t-closeness. We show how these techniques outperform naive approaches and also outline how these approaches can offer privacy to resources.

9.2 Limitations

9.2.1 Static Setting

One of the main limitations of the presented approaches is that we consider anonymization within a static setting. We assumed that one event log of a business process exists and that the data is anonymized at one point in time. Therefore, we do not consider a setting where multiple event logs of one business process exist that might be anonymized at different points in time. Combining different logs could give an adversary information that could not be obtained from one dataset, especially if one individual is involved within several event logs. Even if this is not the case, having already an anonymized log, might offer additional opportunities for better utility-preservation. Furthermore, we also do not tackle the scenario of ongoing business processes, considering the protection of events when they are generated.

9.2.2 Fixed Privacy Parameter

In our studies, we always assumed that a privacy parameter is given and the event log is anonymized to fulfill that guarantee. However, it is also possible to think about settings where, instead, a utility-loss threshold is given and the goal of the anonymization is to provide as much privacy-protection as possible for that threshold. Moreover, it is possible to consider a setting where a specific privacy guarantee shall be given, but a better privacy guarantee could be given with a slightly higher utility loss. Such scenarios are so far not considered in our approaches.

9.3 Future Work

9.3.1 Studying the real-world impact of Anonymization

This thesis and the existing research on privacy-preserving process mining in general aims at providing the highest utility possible for a given privacy guarantee. In this work, we usually measure utility in terms of specific metrics, such as precision and recall for process models. However, so far no study investigated the implications of anonymization on the actual anal-

ysis. It is therefore unclear, how anonymized event logs impact the process analysis and how to best interpret the results drawn from such a log.

9.3.2 Inter-organizational Business Processes

Our anonymization techniques assumed that all the data is present within one event log. If the business process is distributed between several organization the event log might also be distributed between organizations. In such a case, it is unclear how our anonymization techniques can be applied. This kind of distribution is rather common in contexts, such as supply-chains. Therefore, the number of processes our techniques can be applied to is limited. However, the problem of distributed business processes was studied in several other research lines. As an example, the work of Elkoumy et al. [42] considers the construction of a directly-follows-graph that is distributed between several organizations. Similarly, the problem of combining inter-organizational process models has been studied [82].

9.3.3 Privacy-preserving Process Mining beyond Process Discovery

Most anonymization techniques are evaluated in the context of process discovery or through log closeness measures. There is a lack of studies that consider other areas of process mining. Based on the experience obtained in other areas of privacy-preserving data mining, it is probably possible to adjust anonymization techniques developed for general data mining problems for process mining use cases. Therefore, a wide range of potential angles to adjust the existing anonymization techniques for specific process mining techniques exists.

References

[1] Abay, N. C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., and Sweeney, L. Privacy preserving synthetic data release using deep learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18* (2019), Springer, pp. 510–526.

[2] Aggarwal, C. C. On k-anonymity and the curse of dimensionality. In *VLDB* (2005), vol. 5, pp. 901–909.

[3] Aggarwal, C. C. *Data mining: the textbook*. Springer, 2015.

[4] Aggarwal, C. C., and Yu, P. S. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-preserving data mining*. Springer, 2008, pp. 11–52.

[5] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., and Soo, A. Automated discovery of process models from event logs: Review and benchmark. *IEEE transactions on knowledge and data engineering 31*, 4 (2018), 686–705.

[6] Bade, F., Vollenberg, C., Koch, J., Koch, J., and Coners, A. The dark side of process mining. how identifiable are users despite technologically anonymized data? A case study from the health sector. In *Business Process Management - 20th International Conference, BPM 2022, Münster, Germany, September 11-16, 2022, Proceedings* (2022), C. D. Ciccio, R. M. Dijkman, A. del-Río-Ortega, and S. Rinderle-Ma, Eds., vol. 13420 of *Lecture Notes in Computer Science*, Springer, pp. 219–233.

[7] Baier, T., Mendling, J., and Weske, M. Bridging abstraction layers in process mining. *Information Systems 46* (2014), 123–139.

[8] Batista, E., Martínez-Ballesté, A., and Solanas, A. Privacy-preserving process mining: A microaggregation-based approach. *J. Inf. Secur. Appl. 68* (2022), 103235.

[9] Batista, E., and Solanas, A. A uniformization-based approach to preserve individuals' privacy during process mining analyses. *Peer-to-Peer Netw. Appl. 14*, 3 (2021), 1500–1519.

[10] Bauer, M., Fahrenkrog-Petersen, S. A., Koschmider, A., Mannhardt, F., van der Aa, H., and Weidlich, M. 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* (2019), pp. 159–163.

[11] Becker, J., Fischer, R., and Janiesch, C. Optimizing U.S. health care processes - A case study in business process management. In *Reaching New Heights. 13th Americas Conference on Information Systems, AMCIS 2007, Keystone, Colorado, USA, August 9-12, 2007* (2007), J. A. Hoxmeier and S. C. Hayne, Eds., Association for Information Systems, p. 504.

[12] Benarous, M., Toch, E., and Ben-Gal, I. Synthesis of longitudinal human location sequences: Balancing utility and privacy. *ACM Transactions on Knowledge Discovery from Data (TKDD)* (2022).

[13] Berti, A., and van der Aalst, W. M. Reviving token-based replay: Increasing speed while improving diagnostics. In *ATAED@ Petri Nets/ACSD* (2019), pp. 87–103.

[14] Berti, A., van Zelst, S. J., and van der Aalst, W. Process mining for python (pm4py): bridging the gap between process-and data science. *arXiv preprint arXiv:1905.06169* (2019).

[15] Brickell, J., and Shmatikov, V. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), pp. 70–78.

[16] Buijs, J. Receipt phase of an environmental permit application process ('WABO'), CoSeLoG project.

[17] Buijs, J. C., van Dongen, B. F., and van der Aalst, W. M. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems 23*, 01 (2014), 1440001.

[18] Cabanillas, C., Resinas, M., and Cortés, A. R. RAL: A high-level user-oriented resource assignment language for business processes. In *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I* (2011), pp. 50–61.

[19] Camargo, M., Dumas, M., and González, O. Automated discovery of business process simulation models from event logs. *Decis. Support Syst. 134* (2020), 113284.

[20] Carmona, J., van Dongen, B., Solti, A., and Weidlich, M. *Conformance checking*. Springer, 2018.

[21] Chen, R. J., Lu, M. Y., Chen, T. Y., Williamson, D. F., and Mahmood, F. Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering 5*, 6 (2021), 493–497.

[22] Chou, J.-K., Wang, Y., and Ma, K.-L. Privacy preserving event sequence data visualization using a sankey diagram-like representation. In *SIGGRAPH ASIA 2016 symposium on visualization* (2016), pp. 1–8.

[23] Chou, J.-K., Wang, Y., and Ma, K.-L. Privacy preserving visualization: A study on event sequence data. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 340–355.

[24] Cunha, M., Mendes, R., and Vilela, J. P. A survey of privacy-preserving mechanisms for heterogeneous data types. *Computer science review 41* (2021), 100403.

[25] Dankar, F. K., El Emam, K., Neisa, A., and Roffey, T. Estimating the Re-identification Risk of Clinical Data Sets. *BMC Medical Informatics and Decision Making 12*, 1 (2012), 66.

[26] De Koninck, P., vanden Broucke, S., and De Weerdt, J. act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In *International Conference on Business Process Management* (2018), Springer, pp. 305–321.

[27] De Leoni, M. M., and Mannhardt, F. F. Road traffic fine management process, 2015. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[28] de Montjoye, Y.-A., Hidalgo, C. A., Verleysen, M., and Blondel, V. D. Unique in the Crowd: The Privacy Bounds of Human Mobility. *Scientific Reports 3* (2013), 1376.

[29] Desmet, C., and Cook, D. J. Recent developments in privacy-preserving mining of clinical data. *ACM/IMS Transactions on Data Science (TDS) 2*, 4 (2021), 1–32.

[30] Dinur, I., and Nissim, K. 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* (2003), ACM, pp. 202–210.

[31] Djedović, A. Credit Requirement Event Logs. 4TU.Centre for Research Data. Dataset., 2017.

[32] Domingo-Ferrer, J., and Soria-Comas, J. From t-closeness to differential privacy and vice versa in data anonymization. *Knowledge-Based Systems 74* (2015), 151–158.

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

[34] Dumas, M., La Rosa, M., Mendling, J., Reijers, H. A., et al. *Fundamentals of business process management*, vol. 1. Springer, 2013.

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

[36] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference* (2006), Springer, pp. 265–284.

[37] Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), pp. 715–724.

[38] Dwork, C., Smith, A., Steinke, T., and Ullman, J. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application 4*, 1 (2017), 61–84.

[39] Ebert, I., Wildhaber, I., and Adams-Prassl, J. Big data in the workplace: Privacy due diligence as a human rights-based approach to employee privacy protection. *Big Data & Society 8*, 1 (2021), 20539517211013051.

[40] Edmonds, J., and Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM) 19*, 2 (1972), 248–264.

[41] Elkoumy, G., and Dumas, M. Libra: High-utility anonymization of event logs for process mining via subsampling. In *4th International Conference on Process Mining, ICPM 2022, Bolzano, Italy, October 23-28, 2022* (2022), A. Burattin, A. Polyvyanyy, and B. Weber, Eds., IEEE, pp. 144–151.

[42] Elkoumy, G., Fahrenkrog-Petersen, S. A., Dumas, M., Laud, P., Pankova, A., and Weidlich, M. 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* (2020), S. Nurcan, I. Reinhartz-Berger, P. Soffer,

and J. Zdravkovic, Eds., vol. 387 of *Lecture Notes in Business Information Processing*, Springer, pp. 166–181.

[43] Elkoumy, G., Fahrenkrog-Petersen, S. A., Dumas, M., Laud, P., Pankova, A., and Weidlich, M. Shareprom: A tool for privacy-preserving inter-organizational 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), Sevilla, Spain, September 13-18, 2020* (2020), W. M. P. van der Aalst, J. vom Brocke, M. Comuzzi, C. D. Ciccio, F. García, A. Kumar, J. Mendling, B. T. Pentland, L. Pufahl, M. Reichert, and M. Weske, Eds., vol. 2673 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 72–76.

[44] Elkoumy, G., Fahrenkrog-Petersen, S. A., Sani, M. F., Koschmider, A., Mannhardt, F., von Voigt, S. N., Rafiei, M., and von Waldthausen, L. Privacy and confidentiality in process mining: Threats and research challenges. *ACM Trans. Manag. Inf. Syst. 13*, 1 (2022), 11:1–11:17.

[45] Elkoumy, G., Pankova, A., and Dumas, M. 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* (2021), C. D. Ciccio, C. D. Francescomarino, and P. Soffer, Eds., IEEE, pp. 80–87.

[46] Elkoumy, G., Pankova, A., and Dumas, M. Amun: A tool for differentially private release of event logs for process mining (extended abstract). In *Proceedings of the ICPM Doctoral Consortium and Demo Track 2022 co-located with 4th International Conference on Process Mining (ICPM 2022), Bolzano, Italy, October, 2022* (2022), M. Hassani, A. Koschmider, M. Comuzzi, F. M. Maggi, and L. Pufahl, Eds., vol. 3299 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 56–60.

[47] Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security* (2014), pp. 1054–1067.

[48] Fahland, D., and van Der Aalst, W. M. Model repair—aligning process models to reality. *Information Systems 47* (2015), 220–243.

[49] Fahrenkrog-Petersen, S. A. Providing privacy guarantees in process mining. In *Proceedings of the Doctoral Consortium Papers Presented at the 31st International Conference on Advanced Information Systems Engineering (CAiSE 2019), Rome, Italy, June 3-7, 2019* (2019),

M. L. Rosa, P. Plebani, and M. Reichert, Eds., vol. 2370 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 23–30.

[50] Fahrenkrog-Petersen, S. A., Kabierski, M., Rösel, F., van der Aa, H., and Weidlich, M. 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* (2021), C. D. Ciccio, C. D. Francescomarino, and P. Soffer, Eds., IEEE, pp. 72–79.

[51] Fahrenkrog-Petersen, S. A., Kabierski, M., van der Aa, H., and Weidlich, M. Semantics-aware mechanisms for control-flow anonymization in process mining. *Information Systems* (2023), 102169.

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

[53] Fahrenkrog-Petersen, S. A., van der Aa, H., and Weidlich, M. Optimal event log sanitization for privacy-preserving process mining. *Accepted to Data & Knowledge Engineering*.

[54] Fahrenkrog-Petersen, S. A., van der Aa, H., and Weidlich, M. Pretsa: event log sanitization for privacy-aware process discovery. In *2019 International Conference on Process Mining (ICPM)* (2019), IEEE, pp. 1–8.

[55] Fahrenkrog-Petersen, S. A., van der Aa, H., and Weidlich, M. 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* (2020), D. Fahland, C. Ghidini, J. Becker, and M. Dumas, Eds., vol. 12168 of *Lecture Notes in Computer Science*, Springer, pp. 111–128.

[56] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC) 4*, 3 (2001), 224–274.

[57] Fredrikson, M., Jha, S., and Ristenpart, T. 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* (2015), ACM, pp. 1322–1333.

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

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

[60] Guo, Y., Guo, S., Jin, Z., Kaul, S., Gotz, D., and Cao, N. A survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[61] Hintze, M., and El Emam, K. Comparing the benefits of pseudonymisation and anonymisation under the gdpr. *Journal of Data Protection & Privacy 2*, 2 (2018), 145–158.

[62] Holohan, N., Braghin, S., Mac Aonghusa, P., and Levacher, K. Diffprivlib: the ibm differential privacy library. *arXiv preprint arXiv:1907.02444* (2019).

[63] Holohan, N., Leith, D. J., and Mason, O. Optimal differentially private mechanisms for randomised response. *IEEE Transactions on Information Forensics and Security 12*, 11 (2017), 2726–2735.

[64] Kabierski, M., Fahrenkrog-Petersen, S. A., and Weidlich, M. 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* (2021), M. L. Rosa, S. W. Sadiq, and E. Teniente, Eds., vol. 12751 of *Lecture Notes in Computer Science*, Springer, pp. 19–36.

[65] Kabierski, M., Fahrenkrog-Petersen, S. A., and Weidlich, M. Hiding in the forest: Privacy-preserving process performance indicators. *Information Systems 112* (2023), 102127.

[66] Kaczmarek, K., and Koschmider, A. Conceptualizing a log generator for privacy-aware event logs. *EMISA Forum 41*, 1 (2021), 39–40.

[67] Kartal, H. B., Liu, X., and Li, X.-B. Differential privacy for the vast majority. *ACM Transactions on Management Information Systems (TMIS) 10*, 2 (2019), 1–15.

[68] Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. What can we learn privately? *SIAM Journal on Computing 40*, 3 (2011), 793–826.

[69] Kessler, S., Hoff, J., and Freytag, J.-C. Sap hana goes private: from privacy research to privacy aware enterprise analytics. *Proceedings of the VLDB Endowment 12*, 12 (2019), 1998–2009.

[70] Khan, A., Ghose, A., and Dam, H. Cross-silo process mining with federated learning. In *International Conference on Service-Oriented Computing* (2021), Springer, pp. 612–626.

[71] Kirchmann, H., Fahrenkrog-Petersen, S. A., Kabierski, M., van der Aa, H., and Weidlich, M. Privacy-preserving process mining with pm4py (extended abstract). In *Proceedings of the ICPM Doctoral Consortium and Demo Track 2022 co-located with 4th International Conference on Process Mining (ICPM 2022), Bolzano, Italy, October, 2022* (2022), M. Hassani, A. Koschmider, M. Comuzzi, F. M. Maggi, and L. Pufahl, Eds., vol. 3299 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 85–89.

[72] Knols, B., and van der Werf, J. M. E. Measuring the behavioral quality of log sampling. In *2019 International Conference on Process Mining (ICPM)* (2019), IEEE, pp. 97–104.

[73] La Rosa, M., Reijers, H. A., Van Der Aalst, W. M., Dijkman, R. M., Mendling, J., Dumas, M., and García-Bañuelos, L. Apromore: An advanced process model repository. *Expert Systems with Applications 38*, 6 (2011), 7029–7040.

[74] Lavrenovs, A., and Podins, K. Privacy Violations in Riga Open Data Public Transport System. In *AIEEE '16: Proceedings of the IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering* (2016), pp. 1–6.

[75] Leemans, M. NASA Crew Exploration Vehicle (CEV) Software Event Log. TU Eindhoven. Dataset., 2017.

[76] Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers* (2013), pp. 66–78.

[77] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. Mondrian multidimensional k-anonymity. In *22nd International conference on data engineering (ICDE'06)* (2006), IEEE, pp. 25–25.

[78] Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., and Maggi, F. M. Complex symbolic sequence encodings for predictive monitoring of business processes. In *International Conference on Business Process Management* (2016), Springer, pp. 297–313.

[79] Levenshtein, V. I., et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (1966), vol. 10, Soviet Union, pp. 707–710.

[80] Li, N., Li, T., and Venkatasubramanian, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering* (2007), IEEE, pp. 106–115.

[81] Li, T., and Li, N. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), pp. 517–526.

[82] Liu, C., Duan, H., Zeng, Q., Zhou, M., Lu, F., and Cheng, J. Towards comprehensive support for privacy preservation cross-organization business process mining. *IEEE Transactions on Services Computing 12*, 4 (2016), 639–653.

[83] Liu, C., Ge, Y., Xiong, H., Xiao, K., Geng, W., and Perkins, M. Proactive workflow modeling by stochastic processes with application to healthcare operation and management. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014* (2014), S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang, and R. Ghani, Eds., ACM, pp. 1593–1602.

[84] Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *2008 eighth ieee international conference on data mining* (2008), IEEE, pp. 413–422.

[85] Maatouk, K., and Mannhardt, F. Quantifying the re-identification risk in published process models. In *Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers* (2021), J. Munoz-Gama and X. Lu, Eds., vol. 433 of *Lecture Notes in Business Information Processing*, Springer, pp. 382–394.

[86] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 3–es.

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

[88] Mannhardt, F. Sepsis cases-event log. *Eindhoven University of Technology. Dataset* (2016), 227–228.

[89] Mannhardt, F. Multi-perspective process mining. In *BPM (Dissertation/Demos/Industry)* (2018), pp. 41–45.

[90] Mannhardt, F. Responsible process mining. *Process Mining Handbook. LNBIP 448* (2022), 373–401.

[91] Mannhardt, F., and Blinde, D. Analyzing the Trajectories of Patients With Sepsis Using Process Mining. In *RADAR+EMISA '17: Joint Proceedings* (2017), vol. 1859 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 72–80.

[92] Mannhardt, F., Koschmider, A., Baracaldo, N., Weidlich, M., and Michael, J. Privacy-preserving process mining. *Business & Information Systems Engineering 61*, 5 (2019), 595–614.

[93] Mannhardt, F., Petersen, S. A., and Oliveira, M. F. Privacy challenges for process mining in human-centered industrial environments. In *2018 14th International Conference on Intelligent Environments (IE)* (2018), IEEE, pp. 64–71.

[94] Mannhardt, F. (Felix). Hospital billing - event log, 2017.

[95] McCullough, J. S. The adoption of hospital information systems. *Health economics 17*, 5 (2008), 649–664.

[96] McSherry, F., and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* (2007), IEEE, pp. 94–103.

[97] Mendes, R., and Vilela, J. P. Privacy-preserving data mining: methods, metrics, and applications. *IEEE Access 5* (2017), 10562–10582.

[98] Mishra, V. P., and Shukla, B. Process mining in intrusion detection-the need of current digital world. In *International Conference on Advanced Informatics for Computing Research* (2017), Springer, pp. 238–246.

[99] Monreale, A., Pedreschi, D., Pensa, R. G., and Pinelli, F. Anonymity preserving sequential pattern mining. *Artificial intelligence and law 22*, 2 (2014), 141–173.

[100] Munoz-Gama, J., and Carmona, J. A fresh look at precision in process conformance. In *International Conference on Business Process Management* (2010), Springer, pp. 211–226.

[101] Munoz-Gama, J., de la Fuente, R., Sepúlveda, M., and Fuentes, R. Conformance Checking Challenge 2019 (CCC19). 4TU.Centre for Research Data. Dataset., 2019.

[102] Narayanan, A., and Shmatikov, V. Robust De-anonymization of Large Sparse Datasets. In *S&P '08: Proceedings of the 29th IEEE Symposium on Security and Privacy* (2008), pp. 111–125.

[103] Narayanan, A., and Shmatikov, V. De-anonymizing Social Networks. In *S&P '09: Proceedings of the 30th IEEE Symposium on Security and Privacy* (2009), pp. 173–187.

[104] Parameshwarappa, P., Chen, Z., and Koru, G. Anonymization of daily activity data by using ℓ-diversity privacy model. *ACM Trans. Manage. Inf. Syst. 12*, 3 (jun 2021).

[105] Partington, A., Wynn, M. T., Suriadi, S., Ouyang, C., and Karnon, J. Process mining for clinical processes: A comparative analysis of four australian hospitals. *ACM Trans. Manag. Inf. Syst. 5*, 4 (2015), 19:1–19:18.

[106] Party, A. . D. P. W. Opinion 05/2014 on anonymisation techniques. *European Commission* (2014).

[107] Pfitzmann, A., and Köhntopp, M. 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* (2000), H. Federrath, Ed., vol. 2009 of *Lecture Notes in Computer Science*, Springer, pp. 1–9.

[108] Pika, A., Leyer, M., Wynn, M. T., Fidge, C. J., ter Hofstede, A. H. M., and van der Aalst, W. M. P. Mining resource profiles from event logs. *ACM Trans. Manag. Inf. Syst. 8*, 1 (2017), 1:1–1:30.

[109] Pika, A., Wynn, M. T., Budiono, S., Ter Hofstede, A. H., van der Aalst, W. M., and Reijers, H. A. Privacy-preserving process mining in healthcare. *International journal of environmental research and public health 17*, 5 (2020), 1612.

[110] Plaisant, C., and Shneiderman, B. The diversity of data and tasks in event analytics. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis* (2016).

[111] Polyvyanyy, A., Weidlich, M., Conforti, R., Rosa, M. L., and ter Hofstede, A. H. M. The 4c spectrum of fundamental behavioral relations for concurrent systems. In *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings* (2014), G. Ciardo and E. Kindler, Eds., vol. 8489 of *Lecture Notes in Computer Science*, Springer, pp. 210–232.

[112] Rafiei, M., Elkoumy, G., and van der Aalst, W. M. P. Quantifying temporal privacy leakage in continuous event data publishing. In *Cooperative Information Systems - 28th International Conference, CoopIS 2022, Bozen-Bolzano, Italy, October 4-7, 2022, Proceedings* (2022), M. Sellami, P. Ceravolo, H. A. Reijers, W. Gaaloul, and H. Panetto, Eds., vol. 13591 of *Lecture Notes in Computer Science*, Springer, pp. 75–94.

[113] Rafiei, M., Schnitzler, A., and van der Aalst, W. M. P. PC4PM: A tool for privacy/confidentiality preservation in process mining. In *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2021 co-located with 19th International Conference on Business Process Management (BPM 2021), Rome, Italy, September 6th - to - 10th, 2021* (2021), W. M. P. van der Aalst, R. M. Dijkman, A. Kumar, F. Leotta, F. M. Maggi, J. Mendling, B. T. Pentland, A. Senderovich, M. Sepúlveda, E. S. Asensio, and M. Weske, Eds., vol. 2973 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 106–110.

[114] Rafiei, M., and van der Aalst, W. M. P. 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* (2019), C. D. Francescomarino, R. M. Dijkman, and U. Zdun, Eds., vol. 362 of *Lecture Notes in Business Information Processing*, Springer, pp. 676–689.

[115] Rafiei, M., and van der Aalst, W. M. P. 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* (2019), C. D. Francescomarino, R. M. Dijkman, and U. Zdun, Eds., vol. 362 of *Lecture Notes in Business Information Processing*, Springer, pp. 676–689.

[116] Rafiei, M., and van der Aalst, W. M. P. Privacy-preserving data publishing in process mining. In *Business Process Management Forum - BPM Forum 2020, Seville, Spain, September 13-18, 2020, Proceedings* (2020), D. Fahland, C. Ghidini, J. Becker, and M. Dumas, Eds., vol. 392 of *Lecture Notes in Business Information Processing*, Springer, pp. 122–138.

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

[118] Rafiei, M., and van der Aalst, W. M. P. Privacy-preserving continuous event data publishing. In *Business Process Management Forum - BPM Forum 2021, Rome, Italy, September*

06-10, 2021, Proceedings (2021), A. Polyvyanyy, M. T. Wynn, A. V. Looy, and M. Reichert, Eds., vol. 427 of *Lecture Notes in Business Information Processing*, Springer, pp. 178–194.

[119] Rafiei, M., von Waldthausen, L., and van der Aalst, W. M. P. Ensuring confidentiality in process mining. In *Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain, December 13-14, 2018* (2018), P. Ceravolo, M. T. G. López, and M. van Keulen, Eds., vol. 2270 of *CEUR Workshop Proceedings*, CEUR-WS.org, pp. 3–17.

[120] Rafiei, M., von Waldthausen, L., and van der Aalst, W. M. P. Supporting confidentiality in process mining using abstraction and encryption. In *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13-14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers*, P. Ceravolo, M. van Keulen, and M. T. G. López, Eds., vol. 379 of *Lecture Notes in Business Information Processing*. Springer, 2019, pp. 101–123.

[121] Rafiei, M., Wagner, M., and van der Aalst, W. M. Tlkc-privacy model for process mining. In *International Conference on Research Challenges in Information Science* (2020), Springer, pp. 398–416.

[122] Rafiei, M., Wangelik, F., and van der Aalst, W. M. Travas: Differentially private trace variant selection for process mining. *arXiv preprint arXiv:2210.14951* (2022).

[123] Reijers, H. A., van Wijk, S., Mutschler, B., and Leurs, M. Bpm in practice: who is doing what? In *International Conference on Business Process Management* (2010), Springer, pp. 45–60.

[124] Reisig, W. *Petri nets: an introduction*, vol. 4. Springer Science & Business Media, 2012.

[125] Rennie, B. C., and Dobson, A. J. On stirling numbers of the second kind. *Journal of Combinatorial Theory 7*, 2 (1969), 116–121.

[126] Rocher, L., Hendrickx, J., and Montjoye, Y.-A. Estimating the Success of Re-identifications in Incomplete Datasets Using Generative Models. *Nature Communications 10* (2019).

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

[128] Rösel, F., Fahrenkrog-Petersen, S. A., van der Aa, H., and Weidlich, M. A distance measure for privacy-preserving process mining based on feature learning. In *Proceedings of the 17th International Workshop on Business Process Intelligence (BPI).* (2021).

[129] Rozinat, A., and van der Aalst, W. M. Decision mining in prom. In *International Conference on Business Process Management* (2006), Springer, pp. 420–425.

[130] Rozinat, A., and van der Aalst, W. M. P. Conformance checking of processes based on monitoring real behavior. *Inf. Syst. 33*, 1 (2008), 64–95.

[131] Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. *International journal of computer vision 40*, 2 (2000), 99–121.

[132] Senderovich, A., Weidlich, M., Gal, A., and Mandelbaum, A. Queue mining–predicting delays in service processes. In *International conference on advanced information systems engineering* (2014), Springer, pp. 42–57.

[133] Senderovich, A., Weidlich, M., Gal, A., and Mandelbaum, A. Queue mining for delay prediction in multi-class service processes. *Information systems 53* (2015), 278–295.

[134] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. 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* (2017), IEEE Computer Society, pp. 3–18.

[135] Shraga, R., Gal, A., Schumacher, D., Senderovich, A., and Weidlich, M. Process discovery with context-aware process trees. *Information Systems* (2020), 101533.

[136] Smith, J., Asghar, H. J., Gioiosa, G., Mrabet, S., Gaspers, S., and Tyler, P. Making the most of parallel composition in differential privacy. *arXiv preprint arXiv:2109.09078* (2021).

[137] Song, M., Günther, C. W., and Van der Aalst, W. M. Trace clustering in process mining. In *International conference on business process management* (2008), Springer, pp. 109–120.

[138] Song, Y., Dahlmeier, D., and Bressan, S. Not So Unique in the Crowd: a Simple and Effective Algorithm for Anonymizing Location Data. In *PIR@SIGIR '14: Proceeding of the 1st International Workshop on Privacy-Preserving IR* (2014), vol. 2014, pp. 19–24.

[139] Steeman, W. Bpi challenge 2013, 2013.

[140] Stefanini, A., Aloini, D., Benevento, E., Dulmin, R., and Mininno, V. Performance analysis in emergency departments: a data-driven approach. *Measuring Business Excellence* (2018).

[141] Sweeney, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.

[142] Sztyler, T., and Carmona, J. Activities of Daily Living of Several Individuals. University of Mannheim, Germany. Dataset., 2015.

[143] Tassa, T., and Horin, A. B. Privacy-preserving collaborative filtering by distributed mediation. *ACM Trans. Intell. Syst. Technol. 13*, 6 (sep 2022).

[144] Tax, N., Sidorova, N., Haakma, R., and Aalst, W. Mining process model descriptions of daily life through event abstraction. In *Proceedings of SAI intelligent systems conference* (2016), Springer, pp. 83–104.

[145] Tillem, G., Erkin, Z., and Lagendijk, R. L. Privacy-preserving alpha algorithm for software analysis. In *37th WIC Symposium on Information Theory in the Benelux/6th WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux* (2016), pp. 136–143.

[146] Tu, Z., Zhao, K., Xu, F., Li, Y., Su, L., and Jin, D. Protecting trajectory from semantic attack considering k-anonymity, l-diversity, and t-closeness. *IEEE Transactions on Network and Service Management 16*, 1 (2018), 264–278.

[147] van der Aa, H., Leopold, H., and Reijers, H. A. Checking process compliance against natural language specifications using behavioral spaces. *Inf. Syst. 78* (2018), 83–95.

[148] van der Aa, H., Rebmann, A., and Leopold, H. Natural language-based detection of semantic execution anomalies in event logs. *Inf. Syst. 102* (2021), 101824.

[149] Van Der Aalst, W. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS) 3*, 2 (2012), 1–17.

[150] Van der Aalst, W., Weijters, T., and Maruster, L. Workflow mining: Discovering process models from event logs. *IEEE transactions on knowledge and data engineering 16*, 9 (2004), 1128–1142.

[151] van der Aalst, W. M. Intra-and inter-organizational process mining: Discovering processes within and between organizations. In *IFIP Working Conference on The Practice of Enterprise Modeling* (2011), Springer, pp. 1–11.

[152] Van der Aalst, W. M. Extracting event data from databases to unleash process mining. In *BPM-Driving innovation in a digital world*. Springer, 2015, pp. 105–128.

[153] van der Aalst, W. M. Responsible data science: using event data in a "people friendly" manner. In *International Conference on Enterprise Information Systems* (2016), Springer, pp. 3–28.

[154] van der Aalst, W. M. A practitioner's guide to process mining: limitations of the directly-follows graph, 2019.

[155] van der Aalst, W. M. Federated process mining: Exploiting event data across organizational boundaries. In *2021 IEEE International Conference on Smart Data Services (SMDS)* (2021), IEEE, pp. 1–7.

[156] van der Aalst, W. M., and Carmona, J. Process mining handbook, 2022.

[157] van der Aalst, W. M. P. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[158] van der Aalst, W. M. P., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, R. P. J. C., van den Brand, P., Brandtjen, R., Buijs, J. C. A. M., Burattin, A., Carmona, J., Castellanos, M., Claes, J., Cook, J., Costantini, N., Curbera, F., Damiani, E., de Leoni, M., Delias, P., van Dongen, B. F., Dumas, M., Dustdar, S., Fahland, D., Ferreira, D. R., Gaaloul, W., van Geffen, F., Goel, S., Günther, C. W., Guzzo, A., Harmon, P., ter Hofstede, A. H. M., Hoogland, J., Ingvaldsen, J. E., Kato, K., Kuhn, R., Kumar, A., Rosa, M. L., Maggi, F. M., Malerba, D., Mans, R. S., Manuel, A., McCreesh, M., Mello, P., Mendling, J., Montali, M., Nezhad, H. R. M., zur Muehlen, M., Munoz-Gama, J., Pontieri, L., Ribeiro, J., Rozinat, A., Pérez, H. S., Pérez, R. S., Sepúlveda, M., Sinur, J., Soffer, P., Song, M., Sperduti, A., Stilo, G., Stoel, C., Swenson, K. D., Talamo, M., Tan, W., Turner, C., Vanthienen, J., Varvaressos, G., Verbeek, E., Verdonk, M., Vigo, R., Wang, J., Weber, B., Weidlich, M., Weijters, T., Wen, L., Westergaard, M., and Wynn, M. T. Process mining manifesto. In *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I* (2011), F. Daniel, K. Barkaoui, and S. Dustdar, Eds., vol. 99 of *Lecture Notes in Business Information Processing*, Springer, pp. 169–194.

[159] van Dongen, B. Real-life Event Logs - Hospital log. TU Eindhoven. Dataset., 2011.

[160] van Dongen, B. BPI Challenge '12. 4TU.Centre for Research Data. Dataset., 2012.

[161] van Dongen, B. BPI Challenge '15. 4TU.Centre for Research Data. Dataset., 2015.

[162] van Dongen, B. BPI Challenge '17. TU Eindhoven. Dataset., 2017.

[163] van Dongen, B. Bpi challenge 2020, Mar 2020.

[164] van Dongen, B., and Borchert, F. BPI Challenge '18. TU Eindhoven. Dataset., 2018.

[165] van Zelst, S. J., Mannhardt, F., de Leoni, M., and Koschmider, A. Event abstraction in process mining: literature review and taxonomy. *Granular Computing 6*, 3 (2021), 719–736.

[166] Vokinger, K. N., Stekhoven, D. J., and Krauthammer, M. Lost in anonymization—a data anonymization reference classification merging legal and technical considerations. *Journal of Law, Medicine & Ethics 48*, 1 (2020), 228–231.

[167] Von Alan, R. H., March, S. T., Park, J., and Ram, S. Design science in information systems research. *MIS quarterly 28*, 1 (2004), 75–105.

[168] von Voigt, S. N., Fahrenkrog-Petersen, S. A., Janssen, D., Koschmider, A., Tschorsch, F., Mannhardt, F., Landsiedel, O., and Weidlich, M. 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* (2020), S. Dustdar, E. Yu, C. Salinesi, D. Rieu, and V. Pant, Eds., vol. 12127 of *Lecture Notes in Computer Science*, Springer, pp. 252–267.

[169] Wagner, I., and Eckhoff, D. Technical privacy metrics: a systematic survey. *ACM Computing Surveys (CSUR) 51*, 3 (2018), 1–38.

[170] Warner, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association 60*, 309 (1965), 63–69.

[171] Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., and Weske, M. Process compliance analysis based on behavioural profiles. *Information Systems 36*, 7 (2011), 1009–1025.

[172] Weidlich, M., and van der Werf, J. M. E. M. On profiles and footprints - relational semantics for petri nets. In *Application and Theory of Petri Nets - 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings* (2012), S. Haddad and L. Pomello, Eds., vol. 7347 of *Lecture Notes in Computer Science*, Springer, pp. 148–167.

[173] Weijters, A., van Der Aalst, W. M., and De Medeiros, A. A. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP 166* (2006), 1–34.

[174] Wolff, J., and Atallah, N. Early gdpr penalties: Analysis of implementation and fines through may 2020. *Journal of Information Policy 11* (2021), 63–103.

[175] Yang, M., Lyu, L., Zhao, J., Zhu, T., and Lam, K.-Y. Local differential privacy and its applications: A comprehensive survey. *arXiv preprint arXiv:2008.03686* (2020).

[176] Yao, L., Chen, Z., Hu, H., Wu, G., and Wu, B. Privacy preservation for trajectory publication based on differential privacy. *ACM Trans. Intell. Syst. Technol. 13*, 3 (apr 2022).

[177] Yin, X., Zhu, Y., and Hu, J. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Comput. Surv. 54*, 6 (jul 2021).

[178] Yousfi, A., and Weske, M. Discovering commute patterns via process mining. *Knowledge and Information Systems 60*, 2 (2019), 691–713.

[179] Yu, B., Mao, W., Lv, Y., Zhang, C., and Xie, Y. A survey on federated learning in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 12*, 1 (2022), e1443.

[180] Zaman, R., and Hassani, M. Process mining meets gdpr compliance: the right to be forgotten as a use case. In *2019 international conference on process mining doctoral consortium, ICPM-DC* (2019).

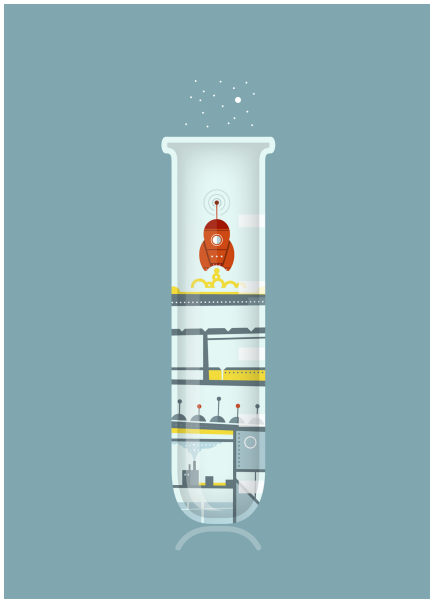
[181] Zhang, J., Borisov, N., and Yurcik, W. Outsourcing security analysis with anonymized logs. In *2006 Securecomm and Workshops* (2006), IEEE, pp. 1–9.

[182] Zhang, L., Wang, W., and Zhang, Y. Privacy preserving association rule mining: Taxonomy, techniques, and metrics. *IEEE Access 7* (2019), 45032–45047.

[183] Zhu, T., Li, G., Zhou, W., and Yu, P. S. Preliminary of differential privacy. In *Differential Privacy and Applications*. Springer, 2017, pp. 7–16.

[184] Zook, M., Barocas, S., Boyd, D., Crawford, K., Keller, E., and Gangadharan, S.P. ... Pasquale, F. Ten Simple Rules for Responsible Big Data Research, 2017.

[185] zur Muehlen, M., and Shapiro, R. Business process analytics. In *Handbook on Business Process Management 2*. Springer, 2010, pp. 137–157.



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, *Science Experiment 02*, was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/suchow/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.

Hiermit erkläre ich, die Dissertation selbstständig und nur unter Verwendung der angegebenen Hilfen und Hilfsmittel angefertigt zu haben. Ich habe mich nicht anderwärts um einen Doktorgrad in dem Promotionsfach beworben und besitze keinen entsprechen- den Doktorgrad. Die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42 am 11. Juli 2018, habe ich zur Kenntnis genommen.

I declare that I have completed the thesis independently using only the aids and tools specified. I have not applied for a doctor's degree in the doctoral subject elsewhere and do not hold a corresponding doctor's degree. I have taken due note of the Faculty of Mathematics and Natural Sciences PhD Regulations, published in the Official Gazette of Humboldt-Universität zu Berlin no. 42 on July 11, 2018.

Berlin, Germany, March 2023

.....

Stephan Fahrenkrog-Petersen