Optimal Event Log Sanitization for Privacy-Preserving Process Mining

Stephan A. Fahrenkrog-Petersen[a,b], Han van der Aa[c], Matthias Weidlich[a]

[a] *Humboldt-Universität zu Berlin, Unter den Linden 6, Berlin, 10117, Berlin, Germany*
[b] *Weizenbaum Institute for the connected Society, Hardenberstr. 32, Berlin, 10623, Berlin, Germany*
[c] *University of Mannheim, Mannheim, 68131, Baden-Württemberg, Germany*

Abstract

Event logs that originate from information systems enable comprehensive analysis of business processes. These logs serve as the starting point for the discovery of process models or the analysis of conformance of a log with a given specification. However, logs potentially contain personal information about individuals involved in process execution. In this paper, we therefore address the risk of privacy attacks on event logs. Specifically, we rely on group-based privacy guarantees instead of noise insertion in order to enable anonymization without adding new behavior to the log. To this end, we propose two new algorithms for event log sanitization that provide privacy guarantees in terms of k-anonymity for the behavioral perspective of a process and t-closeness for sensitive information associated with events. The algorithms thereby avoid the disclosure of employee identities, prevent the identification of employee membership in the log, and preclude the characterization of employees based on sensitive attributes. Our algorithms overcome the limitations of an existing, greedy algorithm, providing users with a trade-off between computational complexity and the utility of the sanitized event log for downstream analysis. Our Experiments demonstrate that sanitization with our algorithms generates event logs of higher utility compared to the state of the art.

Keywords: Event Logs, Data Anonymization, Privacy-preserving Process Mining

1. Introduction

Business processes structure the operations of organizations. They consist of sets of activities that jointly lead to an outcome that is of value to an organization or its clients [12]. Driven by digitization initiatives, business processes are widely supported by IT systems. As a result, even if activities are actually performed by employees, the execution of a business process leaves a trail of data records in the form of event sequences. A collection of such records captured for a particular process is referred to as an *event log*, which provides the starting point for process mining, i.e., the data-driven analysis of business processes [38]: A process model that captures the recorded control-flows of process' activities may be constructed [4]; the conformance of these dependencies may be checked against compliance rules and specifications [25, 26]; and a model may be annotated with performance information to facilitate bottleneck detection and process simulation [18].

Email address: stephan.fahrenkrog-petersen@hu-berlin.de (Stephan A. Fahrenkrog-Petersen)

Since process mining provides valuable insights into their operations, organizations intensify their efforts to record their processes in an accurate and fine-granular manner. Once a process involves manual processing, however, sensitive conclusions based on the resulting event logs are a potential threat. Therefore, such event logs may violate privacy rights [32], such as informational self-determination, i.e., an individual's ability to control, who has access to their personal data [2]. Avoidance of potential privacy breaches is not only an ethical consideration, but is also required by regulations in certain jurisdictions. For example in the European Union the General Data Protection Regulation (GDPR) imposes rules on the processing of personal data unless necessary for a specific purpose [53].

Against this background, event data needs to be privatized to still enable process mining. At first glance, one may resort to pseudonymize the data, i.e. obfuscation to prevent direct identification of entities, to restrict the privacy breaches [45]. However, often such strategies are insufficient, as they do not offer protection against more sophisticated attacks such as frequency-based attacks [23].

Recognizing this, event logs should be sanitized according to well-defined privacy guarantees, such as k-anonymity [51], which ensures that each entity cannot be distinguished from at least k other entities, and ϵ-differential privacy [13], which bounds the impact of each entity on some computation by a privacy budget ϵ. Applying such guarantees to data typically incurs a loss in utility of the data for some analysis question. This loss, in turn, depends on the strength of the guarantee (by choosing k and ϵ) as well as the properties of the data that are preserved by the mechanism to achieve the guarantee, so that combinations of guarantees may turn out to be particularly suitable [49].

In process mining, one important property to consider in data sanitization is behavioural correctness, i.e., the question whether a sanitized event log represents solely behaviour that actually occurred in reality. However, existing techniques that achieve variants of k-anonymity through behavioural suppression [42, 44] and ϵ-differential privacy through noise insertion [20, 21, 31] violate this property: They yield an event log that may contain event sequences that do *not* correspond to actual process executions. While oversampling of event sequences may also be employed to obtain a differentially private log [17], it drastically changes the frequency with which certain behaviour is observed. Either way, a severe bias is created, which may even render an analysis meaningless. For instance, process models constructed from incorrect or uncommon execution dependencies lead to wrong conclusions on the main flow, also perturbing any assessment of performance and resource bottlenecks. Even worse, when verifying the conformance of the event log to a set of compliance rules, mechanisms that do not preserve behavioural correctness potentially lead to harmful conclusions on the involved employees.

In this light, *PRETSA* [19], short for *PRefix-based Event log Sanitization*, has been proposed to prevent *trace linking attacks*, while preserving behavioural correctness. It protects against the correlation of the events of a log with background knowledge, which can result in: (i) *identity disclosure*: whether an event was performed by a specific employee; (ii) *membership disclosure*: whether events of an employee are contained in the log; and (iii) *attribute disclosure*: whether an employee can be characterized through attribute values of events. To this end, PRETSA step-wise transforms a prefix-tree representation of an event log to achieve k-anonymity and its variation for attribute value distributions, i.e., t-closeness. However, the algorithm denotes a greedy strategy that does not yield an optimal

solution, i.e., the loss in utility induced by the transformation may be larger than necessary to achieve the privacy guarantees.

In this article, we generalize the idea underlying PRETSA and phrase prefix-based event log sanitization as a search problem. Based thereon, we contribute two novel algorithms: *PRETSA** instantiates traditional A* search to identify the transformations to apply to yield an event log that satisfies the privacy guarantees, while minimizing the utility loss. The optimality comes at the expense of an exponential worst-case time complexity (in the size of the event log). We therefore also present *BF-PRETSA*, an algorithm that realizes a best-first strategy. While it trades global optimality for runtime performance, it is guided through the search-based formulation of event log sanitization to yield a local optimum of the resulting data utility, thereby improving over the existing PRETSA algorithm. Aside from these improvements to the anonymization of the behavioral perspective, our work also improves the handling of sensitive attributes in event logs. In particular, our work supports the anonymization of these attributes according to stochastic t-closeness, by applying ϵ-differential privacy to the sensitive data, whereas the original PRETSA algorithm could only handle privacy issues of the sensitive attributes through behavior removal.

We evaluate the proposed algorithms in experiments with five real-world datasets. Our results show that PRETSA* drastically improves the data utility of the sanitized event log compared to the PRETSA algorithm. These improvements are well-approximated by BF-PRETSA, which, unlike the optimal PRETSA* algorithms, turns out to show feasible runtimes for realistic instantiations of the sanitization problem.

In the remainder, we first discuss an example scenario (Section 2), before formalizing the context, an attack model, and privacy guarantees (Section 3). We then state the problem of event log sanitisation (Section 4) and propose the algorithms to address this problem (Section 5). Finally, we present evaluation results (Section 6), review related work (Section 7), and conclude the paper (Section 8).

2. Motivation

Lets consider an order handling process, consisting out of activities related to purchase orders (POs) such as their creation (*create_po*), updating (*update_po*), the receiving of goods (*receive_gd*), and activities related to the checking, paying, and rejecting of invoices (*check_in*, *pay_in*, *reject_in*). We assume the respective events of this process are recorded in an event log, where each *trace* consists of a sequence of events executed for a specific process instance. As shown in Table 1, the total of 28 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 2, the durations of the activity executions differ among the traces, even when they are of the same variant.

Privacy violations. Ethical and legal considerations prevent organizations from collecting or disclosing process-related data that compromise the identity of individual employees. Hence, an event log shall not contain information which events are performed by which employee. However, for someone with malicious intent, i.e., an *adversary*, information like the sequence of events may be enough to establish a relation between employees and the execution of certain

Table 1: Trace variants and their counts of an exemplary event log.

ID	Activity sequence (i.e., variant)	Count
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 2: Traces from the exemplary event log, as event sequences with their durations

ID	Event sequence with durations
...	...
v_{21}	*create_po(d:1)*, *update_po(d:10)*, *receive_gd(d:1)*, *check_in(d:5)*, *reject_in(d:2)*
v_{22}	*create_po(d:1)*, *update_po(d:8)*, *receive_gd(d:2)*, *check_in(d:2)*, *reject_in(d:2)*
...	...
v_{31}	*create_po(d:1)*, *receive_gd(d:1)*, *update_po(d:7)*, *check_in(d:25)*, *pay_in(d:3)*
v_{32}	*create_po(d:1)*, *receive_gd(d:3)*, *update_po(d:11)*, *check_in(d:4)*, *pay_in(d:1)*
...	...

events [35]. This, particularly, holds if an adversary has background knowledge from within the organization, 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 employee (*identity disclosure*).
- That the data of a specific employee is included in the event log (*membership disclosure*).
- That an employee can be characterized by execution-related data, e.g. performance data (*attribute disclosure*).

For example, consider a scenario in which goods were received and afterwards some POs have been updated. If we now assume that the adversary has knowledge that that Sue is one of the few employees that has the rights 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 employee that 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. Event log sanitization reduces the probability that such an attack will succeed. To achieve this, event logs are modified in way that ensures that they meet privacy guarantees. A prominent example of these

guarantees [54] is *k-anonymity*, which prevents the disclosure of infrequently occurring process behaviour. In the same vein, the ability to identify employees based on data linked to the execution of an activity may be prevented by notions such as *t-closeness*, which limits the difference between value distributions observed for different equivalence classes.

Consider again the event log from Table 1. An simple strategy to achieve k-anonymity would be to remove all variants from an event log that occurs less than k times. In our example only one variant occurs at least eight times, therefore the sanitized event log would only contain the 10 traces that all represent variant σ_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 σ_{31}.

Event log utility. From a perspective of process mining utility, the major disadvantage of such guarantees is that some useful information may be removed by sanitization. When preserving only the traces that occur at least k times, a significant information loss with regards to the presence and frequency of other sequence variants is introduced. In the aforementioned example, the sanitized log would contain information on only 1 out of the original 5 variants and on only 10 out of 28 traces. Moreover, the removal of attribute values would perturb the statistical basis for process mining. When applying process discovery algorithms on such a sanitized log, the discovered process model would only capture a minority of the actually recorded process behaviour and which may provide misleading results when evaluating the process' performance, e.g., the average cycle time.

It is important to note that a value for k should be selected based on a desired privacy degree and, thus, be independent of the characteristics of an event log at hand. As a result, the impact that a certain k value has on utility will differ per event log. For example, $k = 6$ will have limited impact on a process with thousands of traces and only (relatively) few variants, such as the well-known *Traffic Fines* event logs [10], whereas this impact is much bigger for an event log in which the frequency of most variants is limited, as is the case for the *Sepsis* process [33].

To tackle the issue of utility loss, we propose techniques to reduce the impact of sanitization 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 behavioural correctness, i.e., all traces in the sanitized event log correspond to behaviour that actually occurred in real life. For instance, in Table 1, one can recognize that variants σ_2 and σ_4 are highly similar (only two events are swapped). By transforming the 5 traces of variant σ_4 into ones of σ_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.

3. Event Log Privacy

In the next section, we introduce a formal model. First, we provide a model of of event logs (Section 3.1). Followed by a formal description of the trace linking attack (Section 3.2), and the proposed privacy guarantees to limit the success of the attack (Section 3.3).

3.1. Event Log Model

We will use an event model that is based upon a set of activity identifiers $\mathcal{A}$ (later called activities) and a set of resources $\mathcal{R}$ (in our case, employees). Furthermore, we consider execution-related data, a sensitive attribute of a domain $\mathcal{S}$ that is relevant for the business 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 process analysis may simply be discarded.

We define the universe of events as $\mathcal{E} = \mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$. The universe of event identifiers shall be defined as $\mathcal{I}$. Each of the recorded events $e = (i, a, r, s) \in \mathcal{E}$ represents the recorded execution of a single activity a through a resource r with a corresponding sensitive attribute value s. An event has an unique identifier $i \in \mathcal{I}$, such that for $(i, a, r, s), (i', a', r', s') \in \mathcal{E}$ it holds that $i = i'$ implies that $(i, a, r, s) = (i', a', r', s')$.

A trace is the single execution of a business process. Therefore, it is a finite sequence of events $\sigma = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$. By $\mathcal{T}$, we denote the universe of all traces. A set of traces belonging to the same process form an event log $L = \{\sigma_1, \ldots, \sigma_m\}$ with $\sigma_j \in \mathcal{T}$ for $1 \leq j \leq m$.

All traces that consist of events with the same sequence of activity executions belong to the same variant, as shown in Table 1. Formally, variants induce a partitioning of the event log into disjoint sets of traces $\{V_1, \ldots, V_v\}$, with $V_j \subseteq L$ for $1 \leq j \leq v$ and $L = \bigcup_{1 \leq j \leq v} V_j$.

It is possible to gain information about individual resources from an event log, i.e. the activities executed by a certain employee, , so that it cannot be disclosed directly. Therefore, a projection is considered $\pi : \mathcal{E} \to \mathcal{I} \times \mathcal{A} \times \mathcal{S}$ with $\pi(i, a, r, s) = (i, a, s)$ that removes information on the resource from an event. This projection is lifted to a trace and a log, respectively, by applying it to all contained events, i.e., $\pi(\sigma) = \langle \pi(e_1), \ldots, \pi(e_n) \rangle$ and $\pi(L) = \{\pi(\sigma_1), \ldots, \pi(\sigma_l)\}$.

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 resource are in the log (*membership disclosure*), or on a characterization of resources by values of the sensitive attribute (*attribute disclosure*). However, it might be possible to reveal such information, by incorporating background information.

3.2. Attack Model

We consider a scenario in which a process model is generated from an event log. We assume this model is annotated with performance information that is given by the sensitive attribute $\mathcal{S}$.

Background knowledge. An adversary may possess background knowledge on the relation between resources and activities. In practical business process settings, such information is often available: It may stem, for example, from the

definition of organizational responsibilities (different resources shall execute different sets of activities) [8]; information extracted from shift schedules (resources may differ in when the execute certain sets of activities) [36]; or role-based access control in information systems (different resources can execute different sets of activities) [22].

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 resources, but allows for limiting these assignments based on sequences of activities. Therefore, we can model application contexts that control the assignment of activities to resources in a fine-granular way. In Section 2 we outlined this in the case of a invoice handling process. Let's assume that in this example only specific employees can execute the payment of an invoice (*pay_in*), if the execution of the process is abnormal. Such abnormal behavior could be cases were the purchase order was update (*update_po*) after the goods were already 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 resources $\{r_1, \ldots, r_m\}$. 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 [35] on a project event log. We define its 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 $\textit{work} : \mathcal{R} \to \mathcal{I} \times \mathcal{A} \times \mathcal{S}$, which assigns an event (i, a, s) of a projected trace $\sigma' \in \pi(L)$ to a resource r, such that (i, a, r, s) is an event of the original trace $\sigma \in L$, i.e., $\sigma' = \pi(\sigma)$.
- *Membership disclosure*: To determine that the data of a specific employee is included in a sanitized event log, an adversary aims to identify whether, for a resource $r \in \mathcal{R}$, there exists a projected trace $\sigma' \in \pi(L)$, such that the original trace $\sigma \in L$, $\sigma' = \pi(\sigma)$, 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 an employee can be characterized based on execution-related data, we consider a given distance function over two bags of values of the sensitive attribute, $d : \mathcal{B}(\mathcal{S}) \times \mathcal{B}(\mathcal{S}) \to \mathbb{R}$, 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 resource $r \in \mathcal{R}$, i.e., $d(\sum_{r \in \mathcal{R}} \Gamma(r, a), \Gamma(r, a)) > \psi$ 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 $\lambda : \{1, \ldots, n\} \to \{1, \ldots, m\}$ 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 $\langle a'_1, \ldots, a'_m \rangle$ induces an equivalence class.

For each activity in each equivalence class, the set of resources 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 σ_3 and σ_5. Through the background knowledge 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. Lets consider for the example event log the risk of identify disclosure. We assume at most four invoices could have been paid by one employee, as a consequence at most four events in an equivalence class can relate to one resource. The identity disclosure for one resource, lets say *Jim*, is the construction of $work(Jim)$ and therefore assigning events of the projected log to *Jim*. We can calculate the maximum probability of being successful in this attack. Since the probability is bounded by the ratio of the occurrences of events related to an activity that may have been performed by a specific resource (three) and the total number of events belonging to the equivalence class (six). Consequently, the maximal probability of a successful attack and therefore of assigning the events for invoice payments to *Jim* is $3/6 = 0.5$.

3.3. 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*[51] that can be achieved with regards to identity and membership disclosure. In our setting we define k-anonymity as follows:

Definition 1. *(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 events.*

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 events, consequently k-anonymity provides as a bound on the maximum probability for the success of the aforementioned disclosures.

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

In the aforementioned example, the previously discussed equivalence class consists of eight traces that results in 6-anonymity when only considering the invoice payment (*pay_in*) and yields a bound of $3/6 = 0.5$. 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 $3/12 = 0.25$.

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 resource is part of the

projected log. Nonetheless, this is not possible if each equivalence class has at least k events, so that $k > \tau(a)$ holds true for all activities $a \in \mathcal{A}$. Because in such a setting at least two different resources 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 resource.

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. Therefore, enabling an adversary to generate conclusions linked to the resources 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, the so called called *t-closeness* [30]. 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 $\delta_d : \mathcal{B}(\mathcal{S}) \times \mathcal{B}(\mathcal{S}) \rightarrow \mathbb{R}$ for the respective value distributions. Here, we consider two distance functions proposed in literature:

(1) The Earth Mover's Distance (EMD) [48] 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 [11] 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 operationalization 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 2. *(t-closeness) Let $\pi(L)$ be a projected event log and d 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., $\delta_d(\Omega(a), \Omega'(a)) \leq t$ with $\Omega(a) = [s \mid \langle e_1, \ldots, e_n \rangle \in \pi(L), 1 \leq j \leq n : e_j = (i, a, s)]$ 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 resources involved in its events through the values of the sensitive attribute.

4. The Event Log Sanitization Problem

The previous section defined notions of *k-anonymity* and *t-closeness* as means to provide privacy guarantees for event logs. If a projected event log $\pi(L)$ fulfils both notions, the event log is guarded against trace linking attacks in

accordance with the defined privacy guarantees. However, if $\pi(L)$ does not fulfil these notions, the event log needs to be *sanitized*, i.e., transformed, to fulfil the desired guarantees.

Definition 3 (Event log sanitization)**.** *Let L be an event log and k and t be desired privacy guarantees. Then, event log sanitization is defined as a function $\xi(\pi(L), k, t) = \pi(L)'$, such that the sanitized event log $\pi(L)'$ satisfies the desired k-anonymity and t-closeness privacy requirements.*

As illustrated in Section 2, event log sanitization can considerably affect a log's contents, i.e., the traces, that are present in a sanitized event log $\pi(L)'$. As such, the way in which sanitization is performed directly influences the utility of the resulting event log for process mining tasks. Therefore, event log sanitization shall aim to produce a sanitized event log $\pi(L)'$ that is as close as possible to the original event log $\pi(L)$ as allowed by the desired privacy guarantees.

We quantify the closeness of a sanitized log to the original one by a function that captures the cost that would be required to transform $\pi(L)$ into $\pi(L)'$. Let $\delta_\sigma : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ 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 [47] based on the Act2Vec model [9]. 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. Given a trace distance measure, we define a distance between projected event logs $\pi(L)$ and $\pi(L)'$, as follows:

$$\delta_L(\pi(L), \pi(L)') = \sum_{\substack{\sigma \in \pi(L) \\ \sigma' = \arg\min_{\sigma'' \in \pi(L)'} \delta_\sigma(\sigma, \sigma'')}} \delta_\sigma(\sigma, \sigma'). \tag{1}$$

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

Problem 1 (Event log sanitization problem)**.** *Let L be an event log, k and t be desired privacy guarantees, and δ be a trace cost function. The event log sanitization problem is to derive a sanitized event log $\pi(L)' = \xi(\pi(L), k, t)$, satisfying k-anonymity and t-closeness, such that the distance $\delta_L(\pi(L), \pi(L)')$ is minimal.*

We introduce algorithms that address the event log sanitization problem in the next section.

5. PRETSA-Algorithms Family

Having defined the problem of event log sanitization, 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 and recall the basic PRETSA algorithm (Section 5.1). Subsequently, we introduce PRETSA* (Section 5.2) and *BF-PRETSA* (Section 5.4), two search-based generalizations of the ideas of PRETSA.

Note that, to keep the notation concise, we henceforth assume all event logs to 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.

5.1. Overview

We introduce algorithms to sanitize an event log, i.e., to transform an event log so that it satisfies k-anonymity and t-closeness. As defined in Problem 1, this shall be done while preserving the utility of the event log. The latter requires to keep the distance between the original log and the sanitized log small, ideally minimal.

A first solution to this problem is PRETSA, short for *PRefix-based Event log Sanitization*, proposed by Fahrenkrog-Petersen et al. [19]. PRETSA adopts a prefix-tree representation for an event log, which is step-wise transformed to yield a log that fulfils k-anonymity and t-closeness. The algorithm finds violations to the privacy-constraints by a walk over the prefix-tree. A path in the tree denotes a violation if the number of traces represented at one of its nodes is below the threshold induced by k-anonymity, or if the distance between the value distributions of the sensitive attribute assigned to a node and assigned to whole event log is above the threshold induced by t-closeness. Once such a violation is detected, it is resolved in a greedy manner. The traces corresponding to the violation are merged with the traces of their closest trace variant.

To illustrate the main idea, we take up the traces of Table 1. The prefix-tree representation of the respective event log is shown in Figure 1a. Setting $k = 6$, we detect several violations of k-anonymity. For instance, considering the path corresponding to trace variant σ_2 of Table 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 σ_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 Figure 1b. 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.

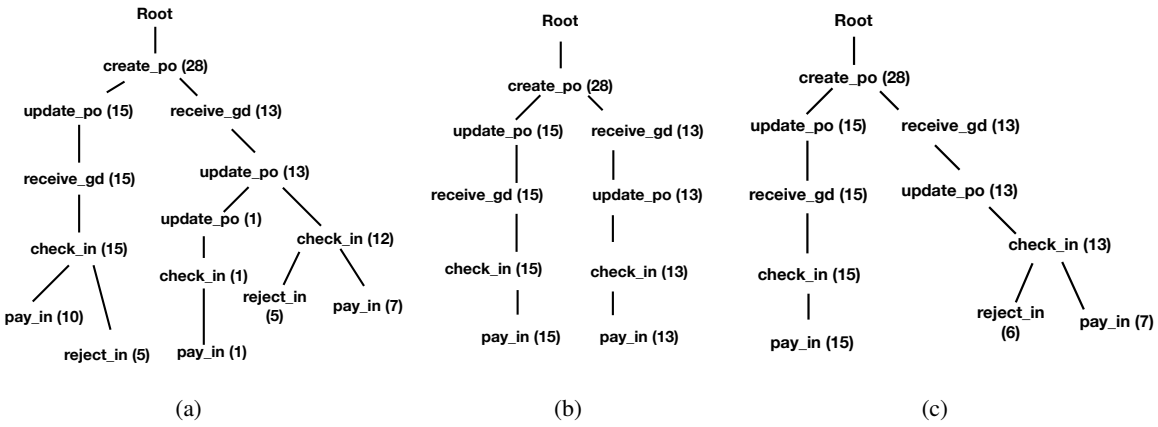

Figure 1: (a) Prefix-tree of the original example log; tree obtained for $k = 6$ with (b) PRETSA and (c) PRETSA*.

While PRETSA achieves event log sanitization in terms of the desired *k-anonymity* and *t-closeness* privacy requirements, it realizes a greedy strategy. It was shown empirically [19] that the utility of logs sanitized by PRETSA

is higher than the utility of those obtained with simple filtering strategies. However, PRETSA does not yield an optimal solution: it only approximates a solution to Problem 1. Striving for optimality, we therefore build upon the ideas of PRETSA and generalize them into two novel algorithms:

PRETSA* formulates event log sanitization 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 original PRETSA algorithm and our new algorithms in Table 3. While all of them guarantee k-anonymity, PRETSA and BF-PRETSA also provide guarantees for either variant of t-closeness, see Section 3.3. PRETSA*, in turn, is limited to the instantiation of t-closeness with the stochastic distance function. The reason being 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 Problem 1. 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 3: Characteristics of the PRETSA algorithms.

Property	**PRETSA [19]**	**PRETSA***	**BF-PRETSA**
k-anonymity	√	√	√
t-closeness variants	EMD, stochastic	stochastic	EMD, stochastic
Data utility	ad-hoc	global optimum	local optimum
Complexity[†]	$O(v^2)$	$O\left(\binom{v}{2}^v\right)$	$O(v^3)$

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

*5.2. PRETSA**

The event log sanitization problem, as formulated in Problem 1, 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 sanitized

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 a sanitized log with minimal distance. Adopting this view, Problem 1 may be approached by a search algorithm, as follows.

Instantiating A*-search for the above setting yields the PRETSA* algorithm, which is defined in Alg. 1. 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 sanitized 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 an log satisfying k-anonymity (i.e., the number of traces is smaller than k).

The above algorithm includes two important design choices. 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 in Section 5.2.1. 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 Alg. 1. We elaborate on the details of this function in Section 5.2.2. Finally, we discuss the time complexity of PRETSA* in Section 5.3.

5.2.1. 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.

Algorithm 1: The PRETSA* algorithm instantiating A*-search for event log sanitization.

input : L, an event log; k and t, privacy parameters; δ_σ, a trace cost function; h, a heuristic to guide the search.

output : L', a sanitized event log.

```
1  open ← {L};                                                    // Set of states to explore
2  closed ← ∅;                                                    // Set of explored states
3  g(L) ← 0;                                                      // Cost of best known path to state
4  f(L) ← g(L) + h(L);                                            // Cost of best known complete path through state
5  while open ≠ ∅ do                                              // While there are states to explore
6      L' ← arg min_{L''∈open} f(L'');                            // Pick the best state according to heuristic
7      closed ← closed ∪ {L'};                                    // Record the state as explored
8      if is_k_anonymous(L', k) then return noisify(L', t) ;      // If k-anonymity is satisfied, add noise
9      for L'' ∈ derive_successor_states(L') do                   // For each successor state
10         if L'' ∈ closed then continue;
11         s ← g(L') + δ_σ(L', L'');                              // Compute cost of new path to state L''
12         if L'' ∉ open then open ← open ∪ {L''};                // Add state to be explored
13         else if s ≥ g(L'') then continue;                      // If new cost is worse, ignore path
14         g(L'') ← s;                                            // Set cost for best known path to state
15         f(L'') ← g(L'') + h(L'');                              // Set cost for best known complete path
16 return ∅;
17 function noisify(L', t)                                        // Procedure to insert noise to achieve t-closeness
18     L ← ∅;                                                     // Log to return
19     for ⟨e_1, ..., e_n⟩ ∈ L' do                                // For each trace
20         σ ← ⟨⟩;                                                // Noisy trace
21         for 1 ≤ j ≤ n, e_j = (i, a, s) do                      // For each event
22             σ ← σ.⟨(i, a, add_noise(s, t))⟩;                   // Add noise to sensitive attribute value
23         L ← L ∪ {σ};                                           // Add noisy trace to log
24     return L;
```

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.

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:

$$h(L') = \sum_{V \in V^-_{L'}} \min(c_{cnv}(V, L'), c_{cvv}(V, L')). \tag{2}$$

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 δ_σ, see Section 4,

from traces to trace variants, i.e., $\delta_V : 2^{\mathcal{T}} \times 2^{\mathcal{T}} \to \mathbb{R}$ and for two trace variants $V, V' \subseteq L'$ the distance is defined as $\delta_V(V, V') = \delta_t(t, t')$ for some $t \in V$ and $t' \in V'$. Then, the cost of merging into a closest non-violating trace variant is:

$$c_{cnv}(V, L') = |V| \cdot \delta_V(V, V') \text{ with } V' = \underset{V'' \in V^+_{L'}}{\arg\min}\, \delta_V(V, V''). \tag{3}$$

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:

$$c_{cvv}(V, L') = \frac{1}{2} \min(|V|, ||V| - k|)\delta_V(V, V') \text{ with } V' = \underset{V'' \in V^-_{L'}}{\arg\min}\, \delta_V(V, V'') \tag{4}$$

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 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.

5.2.2. 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, δ_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 Alg. 1. Here, we exploit the fact that stochastic t-closeness is closely linked to ϵ-differential privacy. As posed by Domingo-Ferrer et al. [11], a differentially private dataset fulfils 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 [11]:

$$t = \max_E \frac{|E|}{N} \left(1 + \frac{N - |E| - 1}{|E|} exp(\epsilon)\right). \tag{5}$$

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:

$$\epsilon = \ln\left(\frac{(\frac{t \cdot N}{|E|} - 1)|E|}{N - |E| - 1}\right) \tag{6}$$

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 sanitization.

Algorithmically, the above idea is formalized in function *noisify* in Alg. 1. 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 sanitized event log.

*5.3. Time Complexity of PRETSA**

PRETSA* constructs a solution to the event log sanitization problem (Problem 1). 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 1. *Given an event log with v trace variants, the time complexity of PRETSA* is given as $O\left(\binom{v}{2}^v\right)$.*

Proof. In general, the A*-search has a time complexity of $O(b^d)$ 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 $v - 1$ merge operations for pairs of trace variants. Now, concerning the branching factor, we note that the number of possible successors in 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 $1/2\binom{v}{k}k^{v-k}$. 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 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 $O\left(\binom{v}{2}^v\right)$. □

Algorithm 2: The BF-PRETSA algorithm, which relaxes PRETSA*.

input : L, an event log; k and t, privacy parameters; δ_σ, a trace cost function; h, a heuristic to guide the search.

output : L', a sanitized event log.

// $g(L), f(L)$ are initialized as in Alg. 1

1 $L_{open} \leftarrow L$; // The currently explored state

2 **while** $L_{open} \neq \emptyset$ **do** // While the current state can be explored further

3 **if** is_k_anonymous(L_{open}, k) **then return** noisify(L_{open}, t) ; // If k-anonymity is satisfied, add noise

4 **if** derive_successor_states(L_{open}) $\neq \emptyset$ **then** // If the current state can be explored further

5 $L_{best} \leftarrow \underset{L'' \in \text{derive\_successor\_states}(L')}{\arg\min} f(L'')$; // Pick the best successor state

6 $g(L_{best}) \leftarrow g(L_{open}) + \delta_\sigma(L_{open}, L_{best})$; // Set cost for best known path to state

7 $f(L_{best}) \leftarrow g(L_{best}) + h(L_{best})$; // Set cost for best known complete path

8 $L_{open} \leftarrow L_{best}$; // Consider the next state to explore

9 **else** $L_{open} \leftarrow \emptyset$; // No further states to explore

10 **return** $\emptyset$;

5.4. BF-PRETSA

PRETSA* solves the problem of event log sanitization, i.e., it transforms 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 1, 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 sanitization 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 the existing PRETSA algorithm. By expanding only the best next candidate state (i.e., event log) in the search space, BF-PRETSA resolves violations of k-anonymity efficiently, i.e., in linear time in the size of the original event log. 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 Alg. 2. 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 Alg. 2. Initially, this log is the original log (line 1). While 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 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 event log sanitization problem. However, the approximate solution can be derived in polynomial time, as discussed next.

Theorem 2. *Given an event log with v trace variants, the time complexity of BF-PRETSA is given as* $O\left(v^3\right)$.

Proof. Considering Alg. 2, 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 1. 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 $O\left({}^1\!/_2(v-1)^2 v\right) \leq O\left(v^3\right)$. □

We note that, with a cubic time complexity, BF-PRETSA is slightly less performant than the original 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 in Section 5.2.1. 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.

6. 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 6.1 we introduce the real-world event logs used in our experiments. The experimantal settings for these experiments is described inSection 6.2, while Section 6.3 discusses the results.

6.1. Datasets

In Table 4 we show the real-world event logs we used to conduct our experiments. 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. The table shows that the employed event logs differ considerably in various key aspects. One very important aspect is the the number of traces per variant, it ranges from an average of 4.3 to 1138.4 over the different event logs. Given that this influencing the performance of the sanitization 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 their number of variants. Since this influences the run time of our algorithms significantly, it enables us to examine the scalability of our solutions.

Table 4: Characteristics of the preprocessed event logs.

				Traces per variant	
Name	**Traces**	**Act.**	**Variants**	**Avg.**	**Max.**
Traffic fines [10]	150,270	11	132	1138.4	56,482
Hospital billing [34]	177,751	17	410	433.5	99,285
CoSeLog [7]	1,348	16	30	44.9	713
BPIC 2013 [50]	1,236	6	76	16.3	485
Sepsis [33]	266	12	62	4.3	35

6.2. Experimental Setup

Algorithms. We compare the results obtained by the three different algorithms of the PRETSA family with each other. Since the evaluation of the original PRETSA algorithm [19] already established that it outperforms simple baselines, we here focus on the comparison of the newly introduced PRETSA* and BF-PRETSA algorithms to the original one. Further, we considered TLKC [42, 44], an anonymization technique is able to consider different kind of background knowledges. 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 deactivated the attribute anonymization of TLKC and focussed only on its control-flow anonymization.

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 implemented the algorithms in a stand-alone Python tool[1]. Our implementation uses AnyTree[2] to implement the Prefix-tree.

All experiments were conducted on a Dell R920 server with an Intel Xeon E7-4880 CPU and 1Tb RAM. We used an execution timeout of 48 hours for each sanitization 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 sanitization procedure:

[1] https://github.com/samadeusfp/PRETSA

[2] https://anytree.readthedocs.io/en/latest/

Log distance: This measure captures the edit distance between the original log L and a sanitized log L'. It is computed as the sum of the edit distance between each trace $t \in L$ and its counterpart in the sanitized 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 sanitization procedure, i.e., the traces for which at least one event was changed, added or deleted by the sanitization 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 sanitization.

Additionally, we also quantify the impact of the sanitization procedure on the utility of a sanitized log L'. We achieve this through two measures, one to assess the control-flow utility and one 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 sanitized log L' compared to the original log L. Specifically, we employ the measure proposed by Knols et al. [29], which considers both the completeness of the relation, as well as the relative frequency with which certain behaviour occurs.

Mean attribute-value error: To quantify the utility of the sanitized attribute values, we also compute the error introduced to the sensitive attribute S in a sanitized 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 sanitization, we assign the maximum error of 1.0.

6.3. Results

This section presents the results of our experiments. Section 6.3.1 first provides insights into the runtime of the algorithms on the real-world event logs, since this is an important distinguishing factor among the algorithms. Then, Section 6.3.2 provides 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, Section 6.3.3 considers the performance of BF-PRETSA for all considered event logs and shows how it outperforms the existing PRETSA algorithm.

6.3.1. Runtime analysis

Figure 2 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, justifying the need for BF-PRETSA.

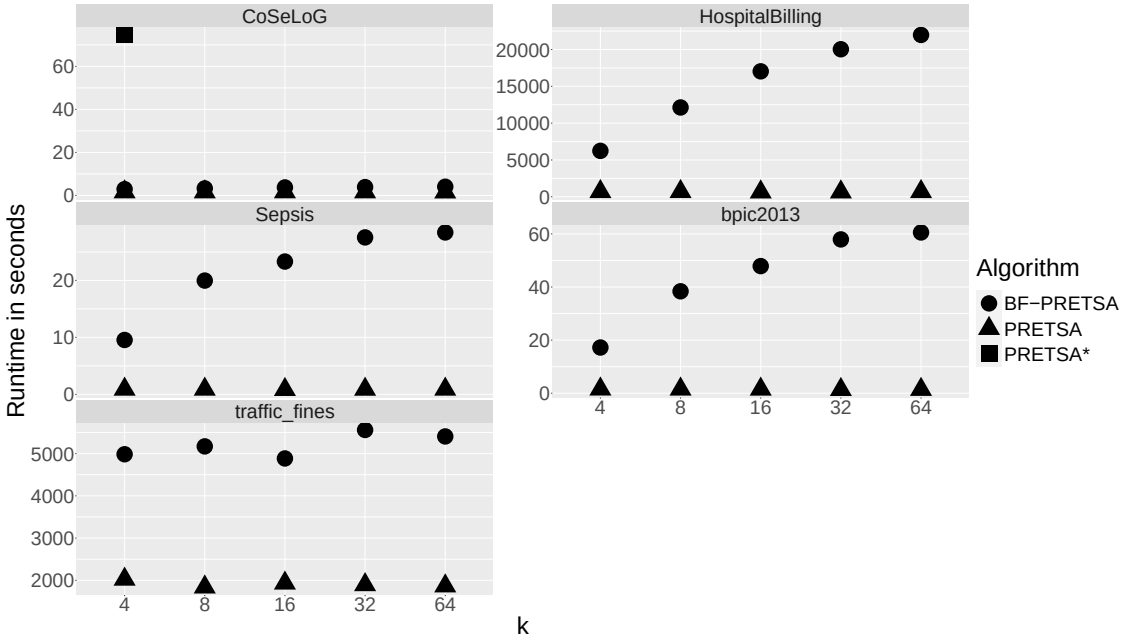

Figure 2: Runtime comparison.

6.3.2. *Comparing PRETSA, BF-PRETSA, and PRETSA**

As shown in Section 6.3.1, 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 5 provides an overview of the main results obtained for these settings. The table reveals that the newly proposed algorithms, BF-PRETSA and PRETSA*, greatly outperform the state of the art, 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 sanitized event log L' much more closely reflects the original log L after applying PRETSA* in comparison to the baseline baseline (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 sanitized 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 behaviour of the sanitized log L' (i.e., the control-flow utility) much closer reflects the behaviour in the original log L after applying PRETSA*. When considering the utility of the sanitized 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 technique only differ in there handling of k-anonymity but not there 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 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

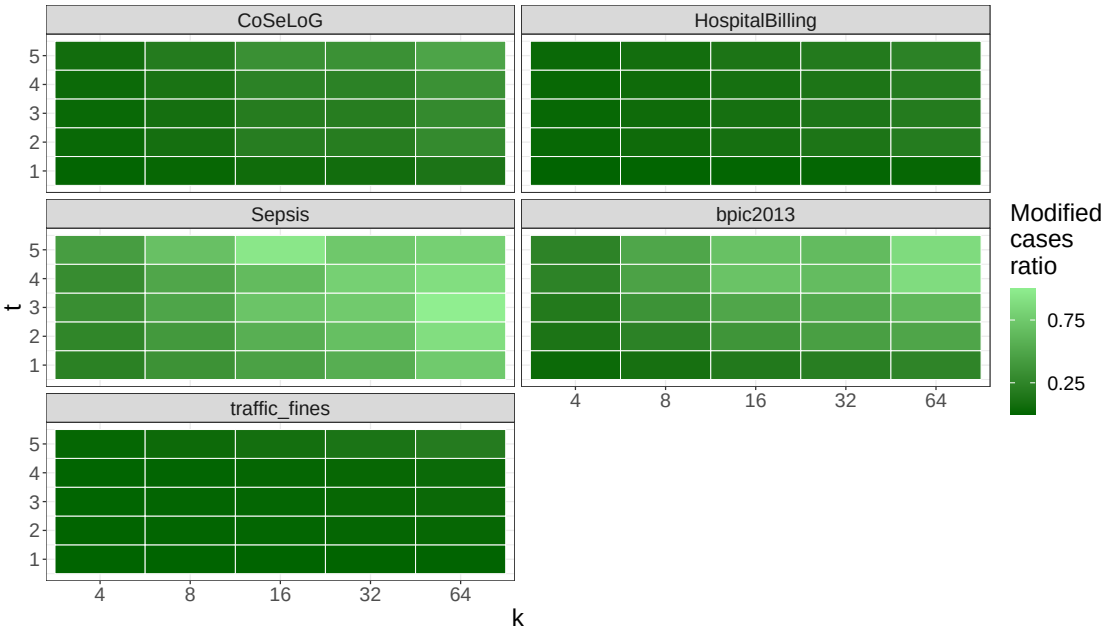

Figure 3: Ratio of modified traces, PRETSA vs BF-PRETSA.

6.3.3. 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 Figure 3. The figure captures the ratio of the traces that are modified by either sanitization 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 Figure 4, 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, Figure 5 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 assumption is supported by Figure 6, which visualizes the increase in percentage points of the directly-follows representativeness obtained BF-PRETSA in comparison with 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

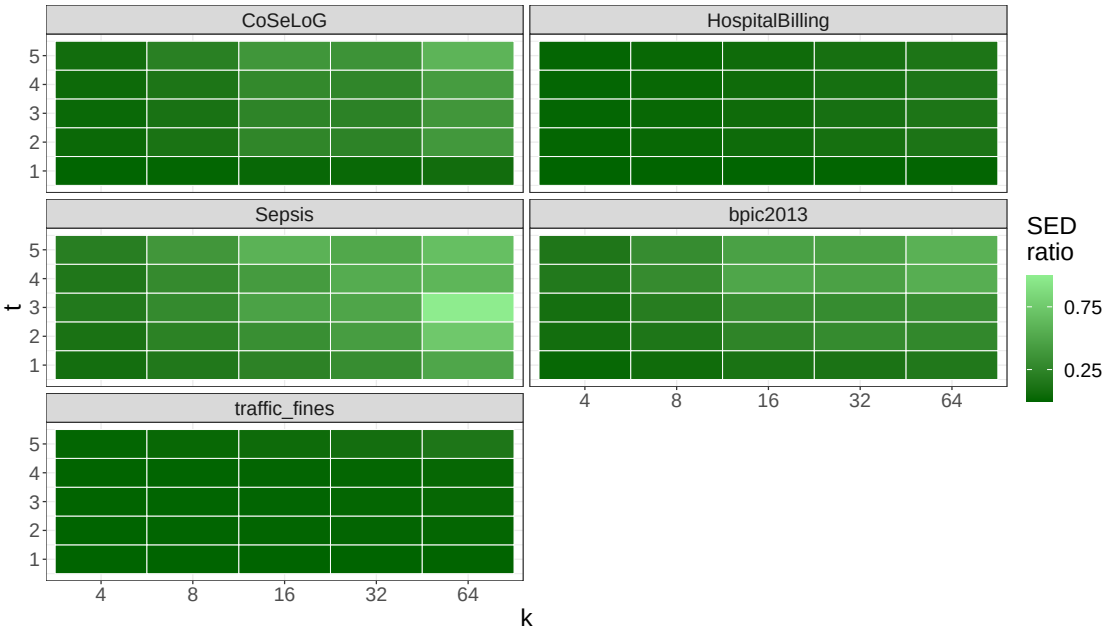

Figure 4: Ratio of edit distance, PRETSA vs BF-PRETSA.

PRETSA for smaller event logs, such as CoSeLoG, Sepsis, and bpics2013, 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 sanitization, as achieved by BF-PRETSA, are more pronounced.

Similar observations are made for the mean attribute-value error. Figure 7 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.

6.3.4. Comparing BF-PRETSA vs. TLKC

We turn to compare BF-PRETSA with the TLKC approach. In Figure 8, we show the number of remaining variants for both techniques for varying values of k. Noteably, for three out of five 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 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 starting activity from all traces. Therefore, BF-PRETSA variants that represent prefixes from the original log provide an additional value. Since, an analyst can be sure 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

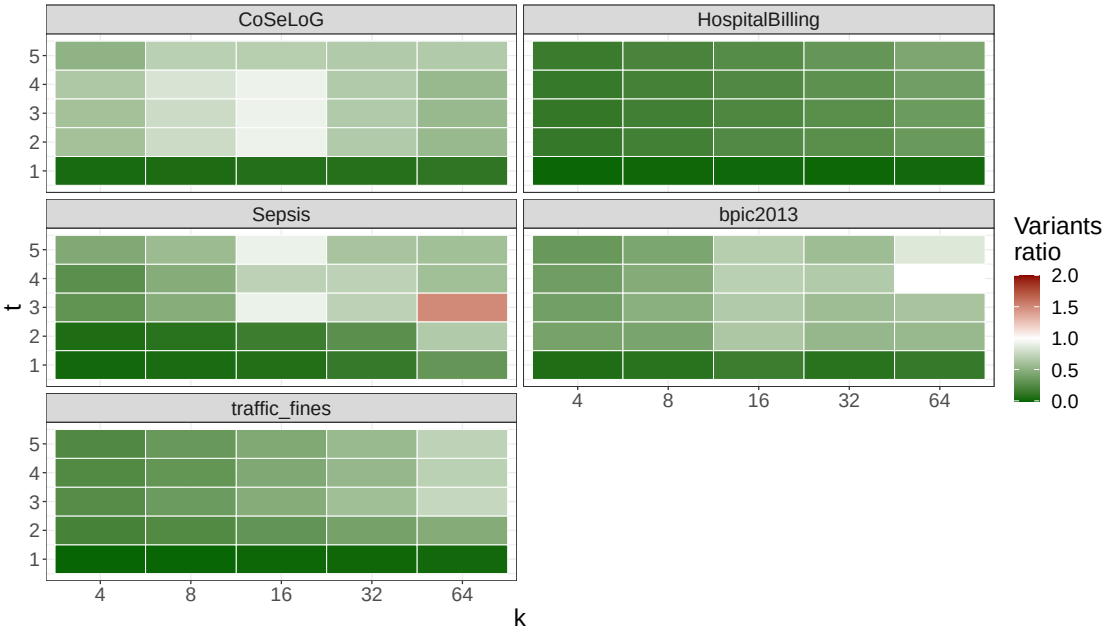

Figure 5: Ratio of retained variants, PRETSA vs BF-PRETSA.

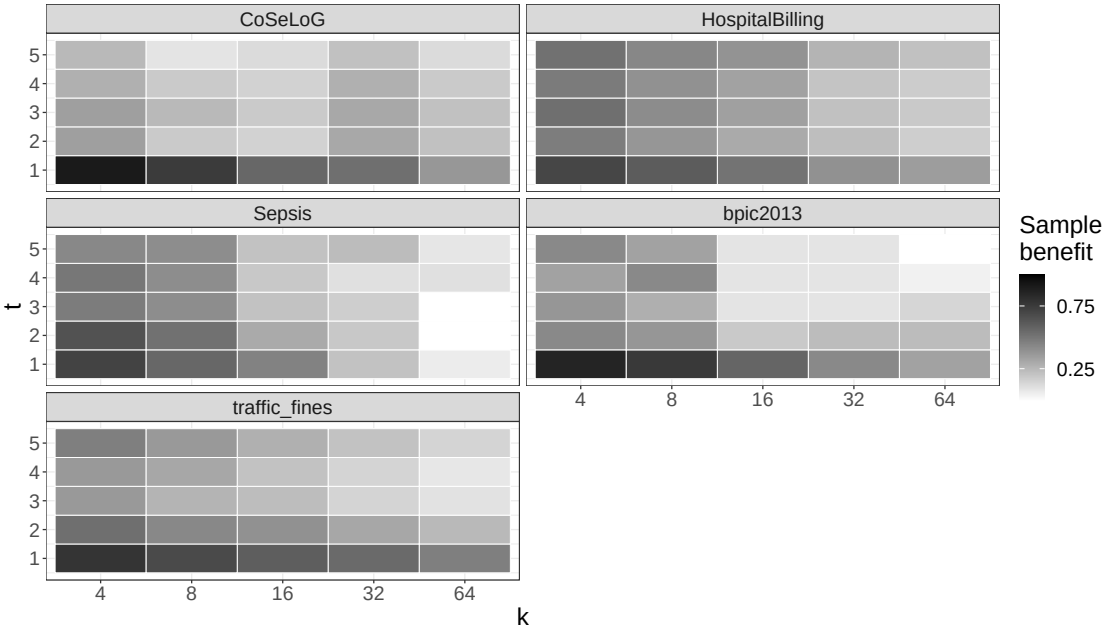

Figure 6: Increase of directly-follows representativeness in percentage points, PRETSA vs BF-PRETSA.

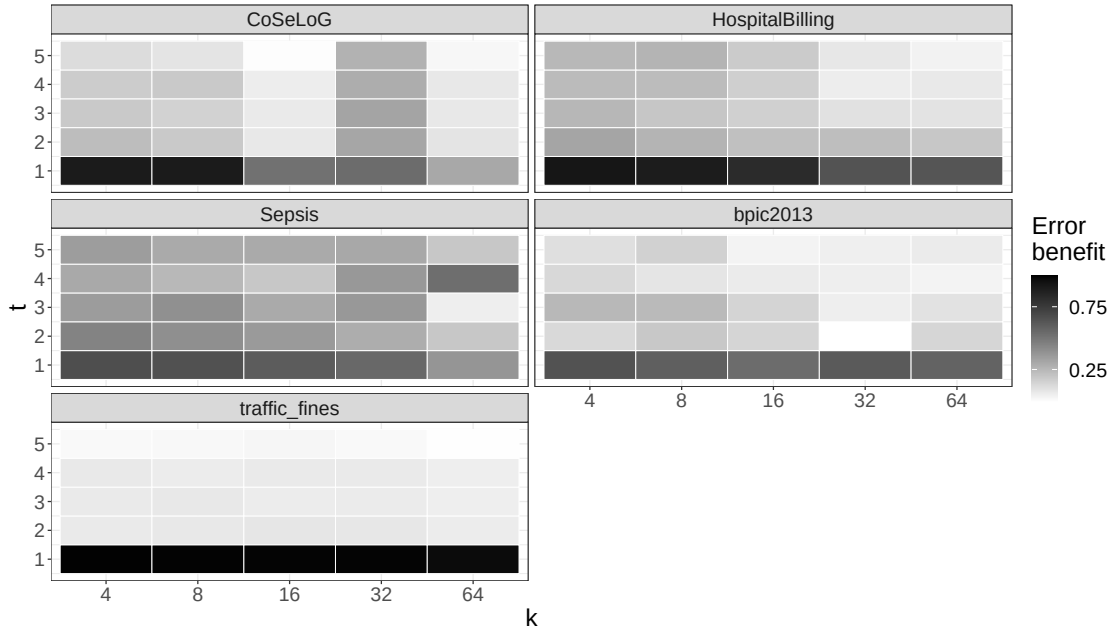

Figure 7: Reduction of mean relative error of cycle time in percentage points, PRETSA vs BF-PRETSA.

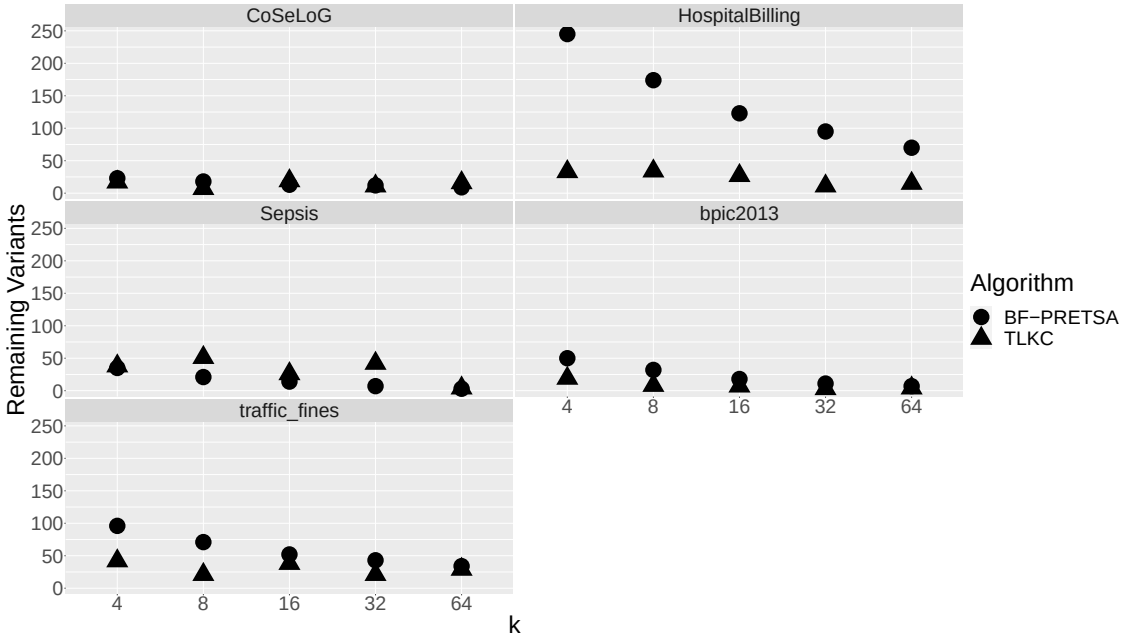

Figure 8: Number of retained variants, TLKC (dotted line) vs BF-PRETSA (straight line).

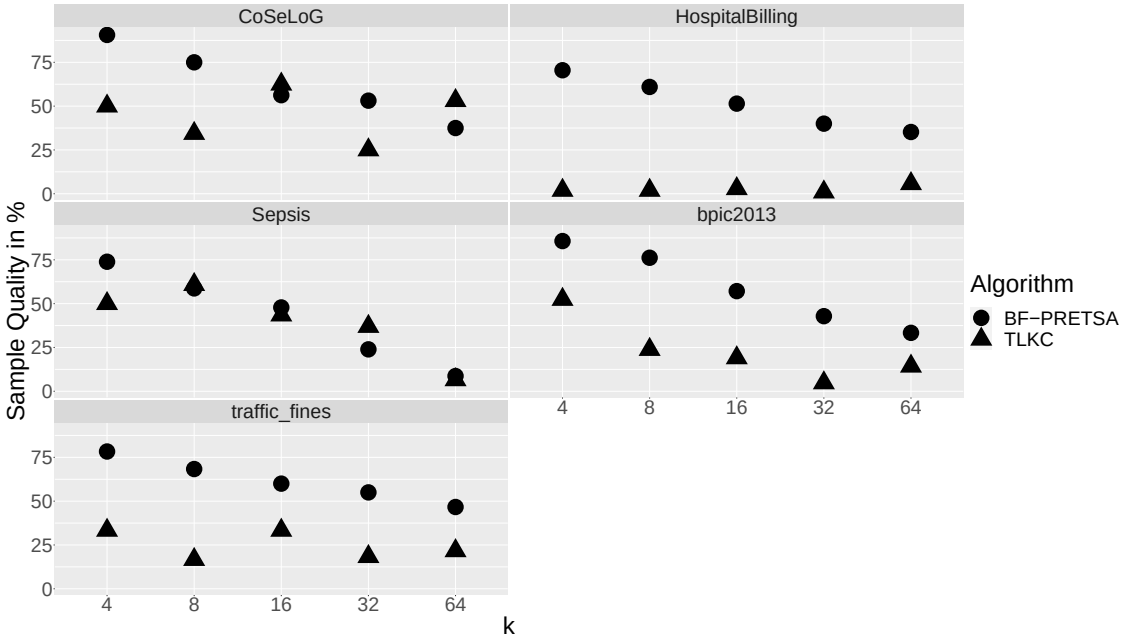

Figure 9: Comparison of directly-follows representativeness in percentage points, TLKC vs. BF-PRETSA.

observe such a benefit for all but one setting, as shown in Table 6.

Table 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

Finally, we compare both techniques in terms of there ability to preserve directly-follows-relations. We show the results in Figure 9. Again, BF-PRETSA outperfoms TLKC on three out of five event logs significantly. In the other two sometimes BF-PRETSA and sometimes TLKC has a slight benefit. Overall, we can conclude that BF-PRETSA can provide significant higher utility that TLKC and seems to never underperform significantly compared to TLKC.

7. Related Work

In the context of process mining the issue of privacy is widely recognized [16, 32, 37]. It was also considered in the broader context of information systems by both academia [24, 1, 52] and industry [28]. In the specific area of

algorithms for k-anonymity several academic tools for privacy in process mining have been released [6, 40] bridging between the academic and practical context.

In terms of specific techniques, work by Rafiei et al. [45] aims to provide confidentiality in process mining through encryption of event log information Batista et al. [5] studies a technique that strengths the privacy protection of pseudonymised and encrypted logs through the uniformization of attribute values within groups of individuals of such logs. Mannhardt et al. [31], in turn, proposed means to realize differentially private queries for the process mining setting. Furthermore, work was aimed at improving the utility of queries [17, 21] or widen the scope of potential queries [27]. Recently, PRIPEL [20] was proposed as a framework that allows the publication of anonymized event logs guaranteeing differential privacy based on so called trace-variant queries [31].

However, with the exception of Elkoumy et al. [17] the above approaches do not guarantee that the published traces represent behavior that was present in the original log and, hence, lack certain utility guarantees given by our approach. Nonetheless, the work by Elkoumy et al. [17] includes all original traces within the anonymized log and therefore revealing even extremely uncommon behavior, a potential angle of attack for an adversary. Other techniques for group-based privacy also do not guarantee that the anonymized behavior was in the original log, such as the techniques proposed as part of the TLKC-privacy framework [44], which delete events from the traces to ensure privacy in terms of relaxations of the traditional k-anonymity guarantee. The algorithms of this framework also employ suppression of events, which can lead to the inclusion of behavior in the anonymized event log that did not appear in the original log. This is a limitation for process mining tasks such as process discovery, which our proposed algorithms are able to avoid. This is a major utility hurdle for process mining tasks such as process discovery, as it was already presented by authors, that their algorithm underperforms the state of the art in important utility metrics for process discovery [3]. Another group-based approach was introduced by Batista et al. [5], based on the uniformization of events within a group of individuals.

Beyond algorithms for event log anonymization, the practical aspects of privacy-aware publishing of event logs have been investigated [41]. Moreover, recently, first approaches for privacy-awareness of event logs in scenarios of continuous data publishing have been studied [43]. Finally, Rafiei et al.[39] introduces a technique that ensure privacy for specific process mining tasks, called role mining. The problem of privacy preservation for inter-organizational process mining was tackled by Zeng et al. [55] and Elkoumy et al. [14, 15]. In a similar way a solutions to distribute the computation of process mining results have been proposed by Rojo et al. [46] that allows for the distribution of different traces independent from each other.

8. Conclusion

In this paper, we focused on the problem of optimal event log sanitization. Taking the existing PRETSA algorithm as a starting point, we generalized the underlying idea and presented *PRETSA** as an algorithm that is based on a search-based formulation of prefix-based event log sanitization. As such, it derives a solution that guarantees

k-anonymity and t-closeness, while preserving the most representative behavior from the original log. In 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 sanitized event logs compared to PRETSA, with BF-PRETSA also showing an acceptable runtime performance.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF), grant number 16DII133 (Weizenbaum-Institute).

References

[1] Abul, O., 2022. Location-privacy preserving partial nearby friends querying in urban areas. Data & Knowledge Engineering 139, 102006.

[2] Asikis, T., Pournaras, E., 2017. Optimization of privacy-utility trade-offs under informational self-determination. CoRR abs/1710.03186.

[3] Augusto, A., Conforti, R., Armas-Cervantes, A., Dumas, M., La Rosa, M., 2020. Measuring fitness and precision of automatically discovered process models: a principled and scalable approach. IEEE Transactions on Knowledge and Data Engineering .

[4] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A., Mecella, M., Soo, A., 2018. Automated discovery of process models from event logs: Review and benchmark. IEEE Transactions on Knowledge and Data Engineering .

[5] Batista, E., Solanas, A., 2021. A uniformization-based approach to preserve individuals' privacy during process mining analyses. Peer-to-Peer Networking and Applications 14, 1500–1519.

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

[7] Buijs, J., 2014. Environmental permit application process ('wabo'), coselog project. doi:10.4121/UUID:26ABA40D-8B2D-435B-B5AF-6D4BFBD7A270.

[8] Cabanillas, C., Resinas, M., Cortés, A.R., 2011. 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, pp. 50–61.

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

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

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

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

[13] Dwork, C., 2008. Differential privacy: A survey of results, in: International Conference on Theory and Applications of Models of Computation, Springer. pp. 1–19.

[14] Elkoumy, G., Fahrenkrog-Petersen, S.A., Dumas, M., Laud, P., Pankova, A., Weidlich, M., 2020a. Secure multi-party computation for inter-organizational process mining , 166–181.

[15] Elkoumy, G., Fahrenkrog-Petersen, S.A., Dumas, M., Laud, P., Pankova, A., Weidlich, M., 2020b. 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, pp. 72–76.

[16] Elkoumy, G., Fahrenkrog-Petersen, S.A., Sani, M.F., Koschmider, A., Mannhardt, F., Von Voigt, S.N.n., Rafiei, M., Waldthausen, L.V., 2021a. Privacy and confidentiality in process mining: Threats and research challenges. ACM Trans. Manage. Inf. Syst. 13.

[17] Elkoumy, G., Pankova, A., Dumas, M., 2021b. Mine me but don't single me out: Differentially private event logs for process mining , 80–87URL: https://doi.org/10.1109/ICPM53251.2021.9576852, doi:10.1109/ICPM53251.2021.9576852.

[18] Estrada-Torres, B., Camargo, M., Dumas, M., García-Bañuelos, L., Mahdy, I., Yerokhin, M., 2021. Discovering business process simulation models in the presence of multitasking and availability constraints. Data & Knowledge Engineering 134, 101897.

[19] Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M., 2019. Pretsa: Event log sanitization for privacy-aware process discovery, in: International Conference of Process Mining.

[20] Fahrenkrog-Petersen, S.A., van der Aa, H., Weidlich, M., 2020. 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, Springer. pp. 111–128.

[21] Fahrenkrog-Petersen, S.A., Kabierski, M., Rösel, F., van der Aa, H., Weidlich, M., 2021. Sacofa: Semantics-aware control-flow anonymization for process mining.

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

[23] Gaines, H.F., 2014. Cryptanalysis: A study of ciphers and their solution. Courier Corporation.

[24] Gharib, M., Giorgini, P., Mylopoulos, J., 2021. Copri v. 2—a core ontology for privacy requirements. Data & Knowledge Engineering 133, 101888.

[25] Herzberg, N., Meyer, A., Weske, M., 2015. Improving business process intelligence by observing object state transitions. Data & Knowledge Engineering 98, 144–164.

[26] Hsu, P.Y., Chuang, Y.C., Lo, Y.C., He, S.C., 2017. Using contextualized activity-level duration to discover irregular process instances in business operations. Information Sciences 391, 80–98.

[27] Kabierski, M., Fahrenkrog-Petersen, S.A., Weidlich, M., 2021. 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, Springer. pp. 19–36.

[28] Kessler, S., Hoff, J., Freytag, J., 2019. SAP HANA goes private - from privacy research to privacy aware enterprise analytics. Proc. VLDB Endow. 12, 1998–2009.

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

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

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

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

[33] Mannhardt, F. (Felix), 2016. Sepsis cases - event log. doi:10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460.

[34] Mannhardt, F. (Felix), 2017. Hospital billing - event log. doi:10.4121/UUID:76C46B83-C930-4798-A1C9-4BE94DFEB741.

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

[36] zur Muehlen, M., Shapiro, R., 2010. Business process analytics, in: Handbook on Business Process Management 2. Springer, pp. 137–157.

[37] Pika, A., Wynn, M.T., Budiono, S., ter Hofstede, A.H., van der Aalst, W.M., Reijers, H.A., 2020. Privacy-preserving process mining in healthcare. International Journal of Environmental Research and Public Health 17, 1612.

[38] Plotnikova, V., Dumas, M., Milani, F.P., 2022. Applying the crisp-dm data mining process in the financial services industry: Elicitation of adaptation requirements. Data & Knowledge Engineering 139, 102013.

[39] Rafiei, M., van der Aalst, W.M., 2019. Mining roles from event logs while preserving privacy, in: International Conference on Business Process Management, Springer. pp. 676–689.

[40] Rafiei, M., van der Aalst, W.M.P., 2020a. Practical aspect of privacy-preserving data publishing in process mining, in: Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2020 co-located with the 18th International Conference on Business Process Management (BPM 2020), Sevilla, Spain, September 13-18, 2020, pp. 92–96.

[41] Rafiei, M., van der Aalst, W.M.P., 2020b. Privacy-preserving data publishing in process mining, in: Business Process Management Forum - BPM Forum 2020, Seville, Spain, September 13-18, 2020, Proceedings, Springer. pp. 122–138.

[42] Rafiei, M., van der Aalst, W.M.P., 2021a. Group-based privacy preservation techniques for process mining. Data Knowl. Eng. 134, 101908.

[43] Rafiei, M., van der Aalst, W.M.P., 2021b. Privacy-preserving continuous event data publishing, in: Business Process Management Forum - BPM Forum 2021, Rome, Italy, September 06-10, 2021, Proceedings, Springer. pp. 178–194.

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

[45] Rafiei, M., von Waldthausen, L., van der Aalst, W.M.P., 2018. 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., pp. 3–17.

[46] Rojo, J., Garcia-Alonso, J., Berrocal, J., Hernández, J., Murillo, J.M., Canal, C., 2022. Sowcompact: A federated process mining method for social workflows. Information Sciences 595, 18–37.

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

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

[49] Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., Martínez, S., 2014. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. The VLDB Journal 23, 771–794.

[50] Steeman, W., 2013. Bpi challenge 2013. doi:10.4121/UUID:A7CE5C55-03A7-4583-B855-98B86E1A2B07.

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

[52] Vatsalan, D., Christen, P., Rahm, E., 2020. Incremental clustering techniques for multi-party privacy-preserving record linkage. Data & Knowledge Engineering 128, 101809.

[53] Voss, W.G., 2017. European union data privacy law reform: General data protection regulation, privacy shield, and the right to delisting. Business Lawyer 72, 221–233.

[54] Wagner, I., Eckhoff, D., 2018. Technical privacy metrics: a systematic survey. ACM Computing Surveys (CSUR) 51, 57.

[55] Zeng, Q., Sun, S.X., Duan, H., Liu, C., Wang, H., 2013. Cross-organizational collaborative workflow mining from a multi-source log. Decision support systems 54, 1280–1301.