PRETSA: Event Log Sanitization for Privacy-aware Process Discovery

Stephan A. Fahrenkrog-Petersen, Han van der Aa, Matthias Weidlich
Humboldt-Universität zu Berlin, Berlin, Germany
Email: {stephan.fahrenkrog-petersen, han.van.der.aa, matthias.weidlich}@hu-berlin.de

***Abstract*—Event logs that originate from information systems enable comprehensive analysis of business processes, e.g., by process model discovery. However, logs potentially contain sensitive information about individual employees involved in process execution that are only partially hidden by an obfuscation of the event data. In this paper, we therefore address the risk of privacy-disclosure attacks on event logs with pseudonymized employee information. To this end, we introduce PRETSA, a novel algorithm for event log sanitization that provides privacy guarantees in terms of k-anonymity and t-closeness. It thereby avoids disclosure of employee identities, their membership in the event log, and their characterization based on sensitive attributes, such as performance information. Through stepwise transformations of a prefix-tree representation of an event log, we maintain its high utility for discovery of a performance-annotated process model. Experiments with real-world data demonstrate that sanitization with PRETSA yields event logs of higher utility compared to methods that exploit frequency-based filtering, while providing the same privacy guarantees.**

1. Introduction

Event logs that are recorded by information systems are the starting point for process mining, i.e., the data-driven analysis of qualitative and quantitative properties of business processes [1]. Specifically, discovery algorithms construct a process model from an event log, thereby formalizing the recorded execution dependencies between a process' activities [2]. Focusing on quantitative properties, such a model is annotated with performance information to facilitate process simulation [3], e.g., to assess the average cycle time.

With the potential of process mining unfolding, organizations intensify their efforts for accurate and fine-granular recording of their processes. Once a process involves manual processing, however, the resulting event logs enable sensitive conclusions on individual employees. As such, event logs may breach privacy [4], violating informal self-determination, i.e., an individual's ability to control, who has access to their personal data [5]. Avoidance of potential privacy breaches is not only an ethical consideration, but may be enforced by legislation. An example is the European General Data Protection Regulation (GDPR) that prohibits processing of personal data unless necessary for a specific purpose [6].

Privacy-aware processing of event logs may be approached based on methods for information security that prevent unauthorized access, use, and disclosure of data [4]. Specifically, data confidentiality may be achieved by restricting the access and interpretation of data by pseudonymization, i.e., the obfuscation of data to prevent direct identification of entities, or anonymization, i.e., the permanent de-identification of data that renders conclusions on entities impossible even when relying on additional data [7].

Such methods, however, are insufficient in terms of privacy in some application contexts as they do not prevent frequency-based attacks [8]. For instance, pseudonymization of employee information in a log does not achieve privacy, if it is known that only a certain employee may execute a specific activity. A different angle therefore is the sanitization of data to give well-defined privacy guarantees. Examples for such guarantees are k-anonymity [9] (each entity cannot be distinguished from at least k other entities) or differential privacy (adding noise to the data, there is a guaranteed level of ambiguity trying to reconstruct the original data) [10].

Many notions of privacy have a parameter that enables fine-tuning of the strengths of the given guarantee, e.g., k in k-anonymity and a privacy budget ϵ in differential privacy. Typically, there is a trade-off between the strengths of a privacy guarantee induced by data sanitization and the loss in utility of the data for some analysis question. For instance, aggregating the events of several instances of a process yields stronger k-anonymity (a higher k value), but perturbs the execution dependencies recorded between activities. Adding more noise to an event log (smaller privacy budget ϵ), yields higher ambiguity in differential privacy, but also decreases the accuracy of process simulation. The actual loss in analysis utility incurred by privacy guarantees depends on the exact definition of respective data sanitization, though. This raises the question of *how to sanitize data such that data utility is maximized under a given privacy guarantee?*

In this paper, we address the above question for a specific application context in process mining. This context is defined by an attack model, which determines the privacy guarantees to consider, and a type of process analysis, which determines how to assess the utility of a sanitized event log.

We consider a *trace linking attack* on an event log with pseudonymized employee information that correlates events with background knowledge on resources. This attack involves (i) identity disclosure, whether an event is related to an 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. We aim to prevent this attack by

sanitizing an event log before it is used to discover a process model, including annotations of activity durations. As such, the utility of the sanitized log is assessed in terms of the change of the model discovered from the sanitized log in comparison to the one discovered from the original log.

For this setting, we present *PREfix-Tree based event log SAnitisation for t-closeness*, shortly *PRETSA*, an event log sanitization algorithm that prevents membership and identity disclosure by k-anonymity and protects against attribute disclosure by t-closeness. In essence, PRETSA constructs a prefix tree representation of an event log that is annotated with frequencies and attribute values. This tree is then stepwise transformed by relocating and merging sub-trees until the required privacy guarantees have been obtained. This way, transformations of the event log are comparatively fine-granular, which implies a modest loss in the log's utility.

We evaluate PRETSA against a baseline that achieves the respective privacy guarantees by filtering the event log. Our experiments with three real-world datasets indicate that the event logs obtained using PRETSA have a high utility for both process discovery and performance analysis. Furthermore, we show that PRETSA outperforms the baseline along various evaluation dimensions and yields good results, even when the baseline fails to provide any results at all.

In the remainder of the paper, we provide a motivating example (Section 2), before formalizing the considered attack model and privacy guarantees (Section 3). We then introduce PRETSA, our algorithm for event log sanitisation (Section 4) and present an empirical evaluation (Section 5). Finally, we review related work (Section 6) and conclude (Section 7).

2. Motivation

To motivate the need for event log sanitization, consider an order handling process, which encompasses activities related to the creation of purchase orders (PO) (*create_po*), updating them (*update_po*), receiving goods (*receive_gd*), and checking, paying, and rejecting invoices (*check_in*, *pay_in*, *reject_in*). Assume that events are recorded for this process, which enables process discovery and performance analysis. While an event log that contains the recorded event data is required to apply these techniques, it is important to recognize that the previously discussed ethical and legal considerations prevent the manager from collecting or disclosing data that compromise the identity of individual employees.

Straightforwardly, this means that an event log must not contain information about which employee performed which events. However, for someone with malicious intent, i.e., an *attacker*, information on the sequencing of events may be enough to relate employees to the execution of certain events [11]. This, particularly, holds if an attacker possess organizational knowledge (e.g., a manager). For instance, a manager may be aware that, for POs that have been updated, only four employees are allowed to check the corresponding invoice. By combining such background knowledge with the traces in an event log, adversaries could 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 some POs have been updated after goods receipt. If an adversary knows that Sue is one of the few employees that are allowed to subsequently check the corresponding invoice, the adversary would be able to identify the specific events that were performed by Sue (identity closure) with high accuracy.

To reduce the probability that such an attack will succeed, event logs must be sanitized to protect the privacy of an organization's employees. One way to achieve this is to alter event logs so that they meet privacy guarantees. An example for such a guarantee is *k-anonymity*, which bars the disclosure of infrequently occurring process behavior. From a process mining perspective, a downside of such a guarantee is that information may become obscured by sanitization.

TABLE 1: Exemplary sequences of activity executions.

	Sequence variant	#
σ_1	*create_po*,*update_po*,*receive_gd*,*check_in*,*pay_in*	10
σ_2	*create_po*,*update_po*,*receive_gd*,*check_in*,*reject_in*	5
σ_3	*create_po*,*receive_gd*,*update_po*,*check_in*,*pay_in*	7
σ_4	*create_po*,*receive_gd*,*update_po*,*check_in*,*reject_in*	5
σ_5	*create_po*,*receive_gd*,*update_po*,*update_po*,*check_in*,*pay_in*	1

Assume that an event log contains events that represent sequences of activity executions as detailed in Table 1. A straightforward manner to ensure k-anonymity, would be to remove any variant from the log that occurs less than k times. Given that only one variant occurs at least eight times, a requirement for k-anonymity with $k = 8$, would yield a sanitized event log that contains only 10 sequences of events that all represent variant σ_1. While this log indeed provides the desired privacy guarantee, it also hides a considerable amount of information on the presence and frequency of other sequence variants. When applying process discovery techniques on this sanitized log, we would therefore discover a process model that only captures a fraction of the actually recorded process behavior.

3. Event Log Privacy

Next, we provide a formal model of the illustrated setting, including a model of event logs (Section 3.1), the considered trace linking attack (Section 3.2), and privacy guarantees to cope with the attack (Section 3.3).

3.1. Event Log Model

To formalize privacy requirements for event logs, we adopt an event model that builds upon a set of activity identifiers $\mathcal{A}$ (activities, for short) and a set of resources $\mathcal{R}$ (i.e., employees). For the execution of an activity by a resource, we further consider execution-related data. Here,

we incorporate this data as a sensitive attribute of a domain $\mathcal{S}$ that is relevant for the analysis. Note that execution-related data that is not important for process analysis does not pose any privacy challenge, as it may simply be discarded.

Let $\mathcal{I}$ be a set of event identifiers. Then, by $\mathcal{E} = \mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$, we denote the universe of events that may be recorded by an information system. As such, each recorded *event* $e = (i, a, r, s) \in \mathcal{E}$ represents an execution of an activity a by a resource r with s as the sensitive attribute value, whereas i is a unique identifier, i.e., 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 single execution of a process, called a *trace*, is a finite sequence of events $t = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$. By $\mathcal{T}$, we denote the universe of all traces. We define an *event log* as a set of traces, $L = \{t_1, \ldots, t_m\}$ with $t_j \in \mathcal{T}$ for $1 \leq j \leq m$.

3.2. Attack Model

We consider a scenario in which a performance-annotated process model shall be discovered from an event log, where performance information is given by the sensitive attribute. Since an event log, as defined above, contains information on individual resources, i.e., the employees that executed a certain activity, it cannot be disclosed. Rather, data that is not relevant for the intended analysis shall be projected. We therefore consider a projection $\pi : \mathcal{E} \to \mathcal{I} \times \mathcal{A} \times \mathcal{S}$ 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., $\pi(t) = \langle \pi(e_1), \ldots, \pi(e_n) \rangle$ and $\pi(L) = \{\pi(t_1), \ldots, \pi(t_l)\}$.

The projected log $\pi(L)$ does not allow for direct conclusions on which activity execution (i.e. which event) was performed by whom (*identity disclosure*), whether events of a resource are part of the log (*membership disclosure*), or on a characterization of resources by values of the sensitive attribute (*attribute disclosure*).

The above information may be revealed, though, if the projected log is combined with background information. Here, we consider such information on the relation between resources and activities. In practice, such information is frequently available: It may stem, for example, from the definition of organizational responsibilities (resources differ in the sets of activities that they are obliged to execute) [12]; role-based access control in information systems (resources differ in the sets of activities that they are allowed to execute) [13]; or information on shift schedules (resources differ in their availability to execute certain sets of activities) [14].

In this work, we incorporate a rather expressive notion of background information that does not only provide insights into possible assignments of activities to resources, but potentially limits these assignments based on activity patterns. This enables us to model application contexts in which the assignment of activities to resources is controlled in a fine-granular manner. As an example, reconsider the invoice handling process of Section 2. In this process, the payment of an invoice may only be performed by specific employees, if the purchase order was previously updated after goods receipt, which indicates an abnormal process execution. We formalize such background information by 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, $b(\langle receive\_gd, update\_po \rangle, pay\_inv) = \{Per, Sue, Amy, Jim\}$ models that *Per*, *Sue*, *Amy*, and *Jim* may pay an invoice of a PO that was updated after goods receipt.

Using such background knowledge, we consider an attack on a projected event log as a specific type of sequence linking attack [11]. That is, given a projected log $\pi(L)$, a *trace linking attack* is an attempt for:

- *Identity disclosure*: Identify a function $work : \mathcal{R} \nrightarrow \mathcal{I} \times \mathcal{A} \times \mathcal{S}$ that assigns an event (i, a, s) of a projected trace $t' \in \pi(L)$ to resource r, such that (i, a, r, s) is an event of the original trace $t \in L$, i.e., $t' = \pi(t)$.
- *Membership disclosure*: For a resource $r \in \mathcal{R}$, identify whether there exists a projected trace $t' \in \pi(L)$, such that the original trace $t \in L$, $t' = \pi(t)$, contains an event (i, a, r, s) for some $i \in \mathcal{I}$, $a \in \mathcal{A}$, and $s \in \mathcal{S}$.
- *Attribute disclosure*: Given a 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 $a \in \mathcal{A}$, identify whether the distribution of values of events related to activity a differs by at least $\delta \in \mathbb{R}$ for some resource $r \in \mathcal{R}$, i.e., $d(\sum_{r \in \mathcal{R}} \Gamma(r, a), \Gamma(r, a)) > \delta$ with $\Gamma(r, a) = [s \mid t = \langle e_1, \ldots, e_n \rangle \in L, 1 \leq j \leq n : e_j = (i, a, r, s)]$.

The above attack is facilitated using background information, as follows. For a projected log $\pi(L)$, the background information b 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 equivalence class, the background information then defines for each activity, the set of resources that may have been involved in the respective events. Taking up the above example (Table 1) and the aforementioned background information, we derive an equivalence class that contains eight traces, i.e., all traces that correspond to variants σ_3 and σ_5. The background information reveals that $\{Per, Sue, Amy, Jim\}$ could have executed the payment of the invoice.

A trace linking attack discloses identity, membership, or attribute values for each of the equivalence classes. We illustrate this for identity disclosure for the above example. Assume that a resource could have paid at most four invoices, i.e., at most four events in the equivalence class can be related to any resource. Picking one resource, say *Sue*, identity disclosure is the construction of $work(Sue)$, assigning events of the projected log to *Sue*. The maximal probability being successful in this construction is bounded by the ratio of the number of events of an activity that may be assigned to a resource (four) and the number of events in the equivalence class (eight). Thus, the maximal probability of correctly assigning events for invoice payments to *Sue* is $4/8 = 0.5$.

3.3. Privacy Guarantees

To reduce the risk that an adversary performs a trace linking attack successfully, the projected event log shall have characteristics that lower the probability of disclosure. This may be achieved for identity and membership disclosure by adopting the notion of *k-anonymity* [9], defined 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.*

As discussed above, the equivalence classes of $\pi(L)$ are induced by the background information b in our model: Each activity pattern of an element of the domain of b defines one equivalence class. By requiring that each class contains at least k events, k-anonymity gives us a direct bound on the maximal probability that disclosure succeeds.

Let $\tau(a)$ be the maximal number of events that can represent 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 $P(e \in work(r)) \leq \tau(a)/k$. Put differently, 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 with a probability of at most $\tau(a)/k$ under k-anonymity.

Taking up the above example, the discussed equivalence class contains eight traces, which corresponds to 8-anonymity when considering only the invoice payment activity and yields a bound of $4/8 = 0.5$. However, if the projected log would satisfy, e.g., 16-anonymity, we would get a tighter bound for the probability of successful disclosure: The maximal probability of correctly assigning events related to the invoice payments to *Sue* would drop to $4/16 = 0.25$.

The above notion of k-anonymity also helps to guard against membership disclosure. Here, we consider the disclosure to be successful if it can be decided with certainty that an event representing an activity execution of a resource is part of the projected log. However, we see that this is not possible if each equivalence class contains at least k events, so that $k > \tau(a)$ for all activities $a \in \mathcal{A}$. In that case, the events in each equivalence class relate to at least two resources, which renders it impossible to conclude with certainty on the association of an event to a specific resource.

Ensuring that the projected log satisfies k-anonymity does not protect against attribute disclosure, though. The values of the sensitive attribute of the events of an activity in an equivalence class may show a distribution that is very different from one over all events of the respective activity, thereby enabling conclusions on the identity of the resources linked to the events in the equivalence class.

To guard against such disclosure, we adopt an enhancement of k-anonymity, called *t-closeness* [15], which limits the amount of information an adversary can gain through the sensitive attribute. Based on [15], with $d : \mathcal{B}(\mathcal{S}) \times \mathcal{B}(\mathcal{S}) \rightarrow \mathbb{R}$ as a distance function (e.g., the Earth Mover's distance [16]), we define this notion as follows:

Definition 2. *(t-closeness) Let $\pi(L)$ be a projected event log and d a distance function. An equivalence class of $\pi(L)$ has t-closeness, if for all activities $a \in \mathcal{A}$, the difference in the value distribution over all events for a in $\pi(L)$, and those in the equivalence class is less or equal than a threshold $t \in \mathbb{R}$, i.e., $d(\Omega(a), \Omega'(a)) \leq t$ with $\Omega(a) = [s \mid t = \langle e_1, \ldots, e_n \rangle \in \pi(L), 1 \leq j \leq n : e_j = (i, a, s)]$ and $\Omega'(a)$ as the restriction of $\Omega(a)$ to events of the equivalence class. $\pi(L)$ has t-closeness, if all its equivalence classes have t-closeness.*

The notion of t-closeness helps 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.

If a projected event log $\pi(L)$ fulfills *k-anonymity* and *t-closeness*, it is guarded against the trace linking attack by the discussed privacy guarantees. If that is not the case, however, the log needs to be *sanitized*. In this work, we consider such a sanitization step as the application of a function f to the event log, which is parametrized by k and t for the desired strengths of the privacy guarantees. It shall yield an event log, $f(\pi(L), k, t) = \pi(L)'$ that adheres to the required privacy guarantees, while preserving utility for process mining.

4. The PRETSA Algorithm

In this section, we introduce the *PRETSA*, an algorithm for *PREfix-Tree based event log SAnitization for t-closeness*. It provides an implementation of the aforementioned sanitization function f for a projected event log. PRETSA is inspired by the BF-P2kA-algorithm (Brute Force Pattern-Preserving k-Anonymization) presented in [11] to sanitize personal data of sequential nature, such as sequences of visited locations. Below, we simplify notation and discuss sanitization of an event log L into a log L', since the projection of resource information ($\pi(L)$ and $\pi(L)'$) is an orthogonal aspect.

With PRETSA, we aim to maximize the utility of a sanitized event log L' for process discovery and process model enhancement. Process discover techniques generally derive *directly follows graphs* (DFG), cf. [17], [18], [19], or similar relations from event logs. In this context, DFGs capture which activities directly succeed each other in an event log's traces and with what frequency. To maximize the utility preserved in a sanitized log L', the DFG that can be constructed from L' should resemble the DFG of the original log L. Therefore, we strive to retain as many of entries of the directly follows relations. In addition, we strive to preserve the quality of activity annotations, used for process model enhancement, by reducing the impact of sanitization on the distribution of values for the sensitive attribute.

To preserve utility, PRETSA transforms an event log into a prefix tree [20]. In a prefix tree, each node is a set of events that all represent executions of the same activity and share the same prefix of activity executions in their respective traces. We capture information about a node as a

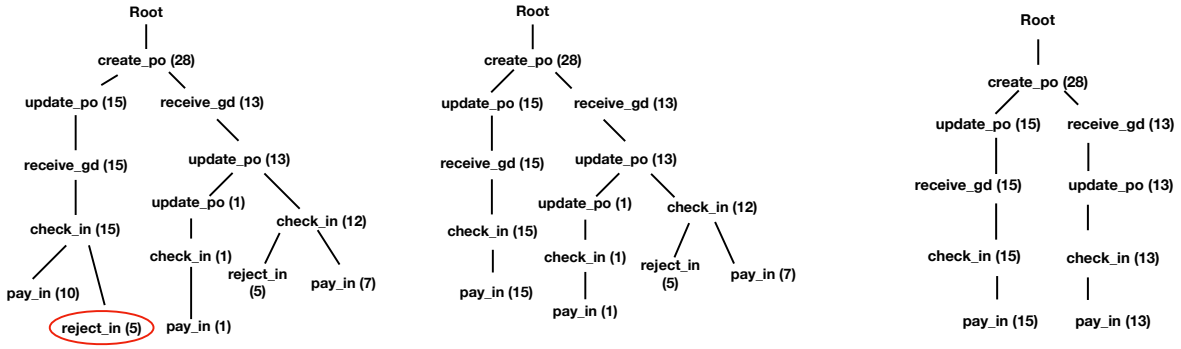

Figure 1: (a) Prefix tree of the example; (b) tree obtained after pruning the highlighted node and reconstructing the tree; (c) final tree returned by PRETSA for the example, setting $k = 8$ and $t = 1.0$.

tuple $n = (a, T, S)$, where $a \in \mathcal{A}$ is the activity of the events; $T \subseteq \mathcal{T}$ is the set of traces that contain the events; $S \in \mathcal{B}(\mathcal{S})$ is the bag of values of the sensitive attribute.

To illustrate the notion of a prefix tree, Figure 1a depicts the tree corresponding to sequences of activity executions in Table 1. For example, the tree shows that the log contains 15 traces that start with the sequence *create_po*,*update_po*,*receive_gd*,*check_in*, that 10 of those are followed by event related to *pay_in*, whereas the remaining five are followed by an event representing activity *reject_in*.

From a privacy perspective, each node in a prefix tree corresponds to a potential equivalence class that may be induced by some background information. In fact, it considers the worst case, in which each prefix of a sequence of executed activities observed in the log defines a separate equivalence class. To fulfill k-anonymity and t-closeness, PRETSA manipulates the prefix tree, such that each node fulfills these properties. The general approach is defined in Alg. 1. We illustrate the algorithm in Figure 1 for the privacy parameters $k = 8$ and $t = 1.0$. The main steps of the algorithm are as follows:

Check privacy guarantees: The algorithm traverses the prefix tree P in a depth-first manner (line 4), until it reaches a node $n = (a, T, S)$ that violates a privacy guarantee (line 5):

- k-anonymity is violated, if $\mid T \mid < k$, i.e., the events of n originate from less than k traces (see Definition 1).
- t-closeness is violated, if $d(\Omega(a), S) > t$ with $\Omega(a)$ as the bag values of the sensitive attribute of all events of activity a of all traces in the log, i.e., if the distance between the overall value distribution and the one at the node is greater than the threshold t (see Definition 2).

For the example in Figure 1a, PRETSA identifies a first violation for the highlighted node. This node violates k-anonymity, given that $\mid T \mid = 5 < k = 8$.

Tree update: If a violation is detected for node $n = (a, T, S)$, we disassociate the traces T from all of n's ancestors (line 6). For instance, pruning the highlighted node in Figure 1a, we remove the five traces that represent the execution sequence *create_po*,*update_po*,*receive_gd*,*check_in*,*reject_in* from the four nodes on the path from the root to the highlighted node. Then, we remove the node and its descendants (if any) from P (line 7).

Algorithm 1 PRETSA(L,k,t)

INPUT: A event log L, the parameter k, a threshold t
OUTPUT: A event log L'

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

Find most similar remaining traces: Next, for each trace $t \in T$ of the pruned node n, PRETSA identifies a trace t' that is most similar (line 8). Here, we consider similarity in terms of the edit (i.e., Levenshtein) distance [21]. For instance, for the highlighted node, the path representing the sequence of activities *create_po*,*update_po*,*receive_gd*,*check_in*,*pay_in*, leading to the far-left leaf node, is most similar (edit distance of one).

Reconstruct tree: Each trace $t \in T$ of the pruned node $n = (a, T, S)$ is then incorporated into all nodes $n' = (a', T', S')$, where T' contains the selected, most similar trace t', which involves adding the events once their activity a has been set to a' (line 9). For all events that have been transformed by PRETSA, the respective values of the sensitive attribute are discarded and replaced by a random value, which is drawn from the distribution of $\Omega(a)$ for events of activity a. For instance, after incorporating the highlighted node into the far-left leaf node, the reconstructed prefix tree will contain 15 traces for the sequence of activity executions *create_po*,*update_po*,*receive_gd*,*check_in*,*pay_in*, rather than the original 10 traces, as shown in Figure 1b.

Termination: The algorithm transforms the prefix tree iteratively, until it is fully traversed without identifying a single violation (line 11). For the running example, the final

tree is shown in Figure 1c. Based on the obtained prefix tree, PRETSA returns a sanitized event log as the set of all traces represented by the tree (line 12).

Applying PRETSA to our example, with $k = 8$ and $t = 1.0$, we end up with the sanitized event log that contains traces of the following sequences of activity executions:

create_po,update_po,receive_gd,check_in,pay_in	15
create_po,receive_gd,update_po,check_in,pay_in	13

A naive approach that deletes all traces that violate the privacy requirements would yield a less representative log:

create_po,update_po,receive_gd,check_in,pay_in	10

5. Evaluation

This section presents an experimental evaluation of the PRETSA algorithm. We show that PRETSA enables the sanitization of event logs, while preserving utility for process mining. Section 5.1 introduces the three real-world event logs used in our experiments, Section 5.2 describes the experimental setup, while Section 5.3 discusses the results.

5.1. Datasets

We conducted our evaluation experiments based on the three real-world event logs characterized in Table 2. The table shows that the characteristics of the event logs differ considerably. Most notably the number of cases per variant ranges from an average of 1.2 to 651.0. Given that this factor is crucial to the performance of a sanitization approach, we believe that the utilized data collection is well-suited to achieve a high external validity of the results.

TABLE 2: Characteristics of the utilized event logs

Name	Cases	Variants	Cases per variant Avg.	Cases per variant Max.
Traffic Fines [22]	150,370	231	651.0	56,482
CoSeLog [23]	1,434	116	12.4	713
Sepsis [24]	1,050	846	1.2	35

5.2. Experimental Setup

To conduct our experiments, we implemented PRETSA as a stand-alone Python program that is available on Github[1]. We use this implementation to analyze two aspects:

1) Impact of k-anonymity on discovered models. Guaranteeing k-anonymity can affect the quality of process models discovered from sanitized event logs. To assess this, we first discover a process model from a sanitized event log L' using the Inductive Miner infrequent [19]. Afterwards, we quantify the quality of this discovered model by determining its *fitness* [25] and *precision* [26] with respect to the original event log L. In this way, we show how accurately the discovered process models discovered capture the behavior of the original non-sanitized log. In our experiments, we assess the impact of various k values, using $k = \{2, 4, 8, 16, 32, 64, 128, 256\}$.

2) Impact of t-closeness on model enhancement. Guaranteeing t-closeness can affect the accuracy of process model annotations derived from sanitized event logs. To determine this impact, we compare annotations, in particular execution times, derived from an event log L', sanitized by PRETSA, to annotations derived from the original event log L. If an event log did not contain an end timestamp for an activity, we generate the execution time of an event e_i as the time difference between the start time of the event e_i and the start time of the following event e_{i+1}. In our experiments, we assess the impact of various k values, using $t = \{0.025, 0.050, 0, 075, 0.100\}$.

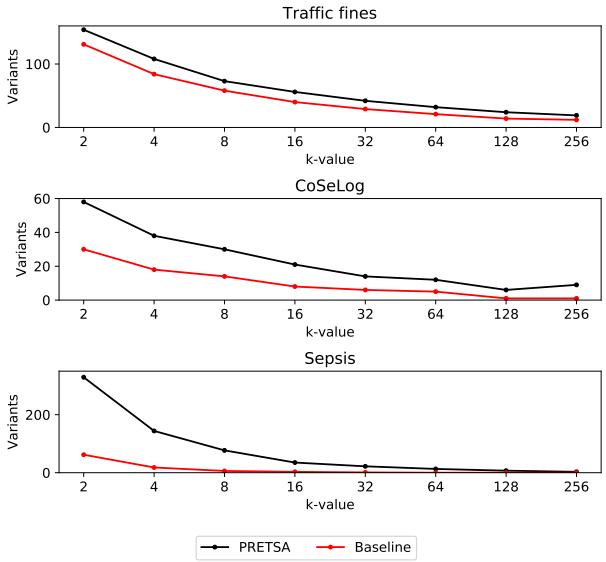

Figure 2: Number of variants retained for k-anonymity.

We compare the results obtained using PRETSA to a baseline approach. For this baseline, we provide privacy guarantees in the following, rudimentary manner:

k-anonymity baseline. To guarantee k-anonymity for the baseline log L'_{BL}, we remove traces that represent an activity sequence that occurs less than k-times in L.

t-closeness baseline. To guarantee t-closeness for the baseline log L'_{BL}, we compare the distribution of the execution times for an activity $a \in \mathcal{A}$ when it is part of traces of a specific sequence variant to the overall distribution of the throughput times for activity a in the original log L. If these two distributions statistically differ, then the t-closeness requirement is not met. We then discard all traces of the respective sequence variant in the construction of log L'_{BL}.

5.3. Results

Impact of k-anonymity on discovered models. Figure 2 shows how the contents of event logs are preserved for varying k values. We see that, for both PRETSA and the

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

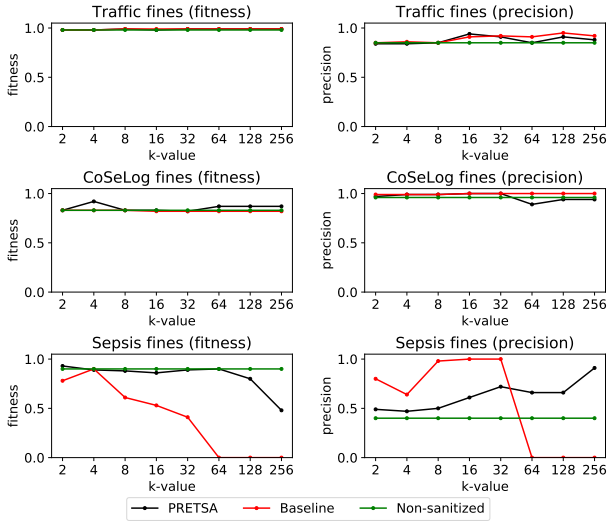

Figure 3: Fitness and precision results based on the Inductive Miner infrequent (default parameters).

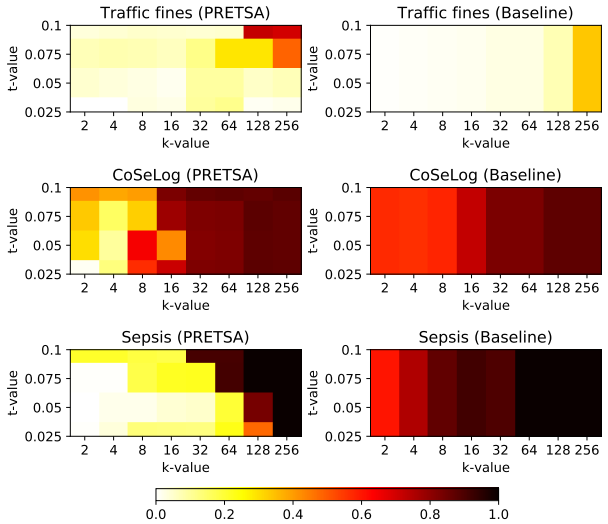

Figure 4: Distance between annotations in sanitized versus non-sanitized event logs (with 0.0 as equal annotations).

baseline, the number of contained variants considerably decreases for higher values of k. However, using PRETSA, we are able to consistently retain more variants than the baseline. For instance, with $k = 4$, log L'_{BL} contains just 18 variants for the Sepsis case, whereas the log obtained by PRETSA still contains 144 variants. For this case, with $k \geq 64$ the baseline log actually contains zero variants, because the most common trace occurs only 35 times. PRETSA on the other hand always retains at least 3 variants.

Similar trends can be observed in Figure 3, in which we depict the log fitness and precision obtained when comparing an original event log L to the process models discovered by applying the Inductive Miner on the sanitized event logs. This figure also depicts the results obtained when discovering models based on non-sanitized event logs.

For the *traffic fines* case, the figure shows that the fitness and precision values do not differ considerably across the three approaches. This outcome is in line with the expectations for this case, because most variants in the log are rather common. As a result, even a naive sanitization approach, i.e., the baseline, has a limited impact on the results. For the *CoSeLog* case, we observe that PRETSA outperforms the baseline for most k-values, in terms of log fitness, but may lead to smaller precision values. The biggest differences among the approaches can be observed for the *Sepsis* case, because the baseline discards a considerable number of variants. For instance, for $k = 64$, PRETSA still achieves a fitness of 0.90, whereas the baseline yields a null result. This demonstrates that PRETSA is particularly useful for less-structured processes.

Impact of t-closeness on model enhancement. Figure 4 presents heatmaps that depict the impact of sanitization for t-closeness on the accuracy of execution time annotations. Given specific k and t values, the figure indicates how close the average execution times of activities in the sanitized log L' are to the original average in the original logs. In particular, we represent the distance between the two averages, where a distance of 0.0 means that they are equal. Overall, PRETSA is shown to have a higher annotation accuracy than the baseline. However, the results across the three cases differ.

For the *traffic fines* case, we observe that both PRETSA and the baseline provide accurate annotations. As noted earlier, this is due to the high number of traces per sequence variant in this log. Consequently, privacy guarantees can be provided with limited impact on utility. For the *CoSeLog* case, there are considerable differences between the original annotations and the annotations derived from sanitized event logs. For $k > 8$, the accuracy drops, as an increasing number of events are omitted, although PRETSA still achieves better accuracy than the baseline. These observations are interesting, because we previously showed that the impact of k-anonymity on the quality of discovered process models is rather limited for this case. Finally, for the *Sepsis* case, we observe clear differences between the performance of PRETSA and the baseline. For $k \leq 32$, the annotations of PRETSA are highly accurate. By contrast, the baseline approach struggles to provide accurate annotations for any k and t.

Overall, our evaluation experiments show that PRETSA outperforms the rudimentary baseline methods in providing privacy guarantees in most evaluation settings. However, the results also illustrate that strong privacy requirements impact the quality of results obtained by process model discovery and enhancement techniques.

6. Related Work

The presented work relates to the consideration of privacy in the contexts of process and sequence mining.

The importance of privacy has been widely recognized in the area of information systems engineering, cf., [27], [28]. Recently, privacy concerns have been acknowledged in the specific context of process mining, for instance by Van der Aalst [29]. Mannhardt et al. [4] provide an overview of privacy challenges and guidelines in this regard, covering

both technological as well as organizational aspects. Work by Rafiei et al. [7] aims to provide confidentiality in process mining through encryption of event log information. However, this approach does not provide any privacy guarantees.

In the context of sequence mining, Monreal et al. [11] provide an approach that achieves k-anonymity for sequential data, which served as basis for the PRETSA algorithm proposed in our work. Crucially, the work by Monreal et al. does not consider data payload, which is required in order to prevent attribute disclose attacks, such as achieved by PRETSA through t-closeness guarantees. Moreover, the iterative, step-wise nature of PRETSA allows it to preserve more event log information.

7. Conclusion

In this paper, we considered how to provide privacy guarantees for events logs while preserving data utility for process mining analyzes. For this purpose, we introduced a model for privacy-disclosure attacks that utilizes execution sequences to derive sensitive information. To counter such attacks, we developed the PRETSA algorithm to sanitize event logs while providing privacy guarantees in terms of k-anonymity and t-closeness. PRETSA preserves utility for process mining through step-wise transformation of the prefix-tree representation of an event log. An experimental evaluation using three real-world data sets demonstrates that PRETSA indeed achieves this goal: the evaluation results show that sanitization with PRETSA yields event logs of higher utility compared to a baseline using frequency-based filtering, while providing the same privacy guarantees.

In future research, PRETSA may be extended so that it avoids reaching local optima and instead converges into a global optimum, in order to preserve more event log information. Moreover, we strive to develop sanitization techniques that provide even stronger privacy guarantees, in the form of *differential privacy* [10].

Acknowledgments. Parts of this research were funded by the Alexander von Humboldt Foundation.

References

[1] W. M. Van der Aalst, *Process mining: data science in action*. Springer, 2016.

[2] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. M. Maggi, A. Marrella, M. Mecella, and A. Soo, "Automated discovery of process models from event logs: Review and benchmark," *TKDE*, in print, 2018.

[3] W. M. P. van der Aalst, "Business process simulation survival guide," in *Handbook on BPM 1*. Springer, 2015, pp. 337–370.

[4] F. Mannhardt, S. A. Petersen, and M. F. Oliveira, "Privacy challenges for process mining in human-centered industrial environments," in *IE*. IEEE, 2018, pp. 64–71.

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

[6] W. G. Voss, "European union data privacy law reform: General data protection regulation, privacy shield, and the right to delisting," *Business Lawyer*, vol. 72, no. 1, pp. 221–233, 2017.

[7] M. Rafiei, L. von Waldthausen, and W. M. P. van der Aalst, "Ensuring confidentiality in process mining," in *SIMPDA*, ser. CEUR Workshop Proceedings, vol. 2270. CEUR-WS.org, 2018, pp. 3–17.

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

[9] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int'l Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[10] C. Dwork, "Differential privacy: A survey of results," in *TAMC*. Springer, 2008, pp. 1–19.

[11] A. Monreale, D. Pedreschi, R. G. Pensa, and F. Pinelli, "Anonymity preserving sequential pattern mining," *Artificial intelligence and law*, vol. 22, no. 2, pp. 141–173, 2014.

[12] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Ral: A high-level user-oriented resource assignment language for business processes," in *BPM Workshops*. LNBIP, vol. 99. Springer, 2011, pp. 50–61.

[13] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM TISSEC*, vol. 4, no. 3, pp. 224–274, 2001.

[14] M. zur Muehlen and R. Shapiro, "Business process analytics," in *Handbook on BPM 2*. Springer, 2010, pp. 137–157.

[15] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *ICDE*. IEEE, 2007, pp. 106–115.

[16] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[17] A. Augusto, R. Conforti, M. Dumas, and M. La Rosa, "Split miner: Discovering accurate and simple business process models from event logs," in *ICDM*. IEEE, 2017, pp. 1–10.

[18] A. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in *CIDM*. IEEE, 2011, pp. 310–317.

[19] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *BPM Workshops* , LNBIP, vol. 171. Springer, 2013, pp. 66–78.

[20] R. De La Briandais, "File searching using variable length keys," in *Western joint computer conference*. ACM, 1959, pp. 295–298.

[21] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[22] M. De Leoni and F. Mannhardt, "Road traffic fine management process," 2015. https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

[23] J. Buijs, "Environmental permit application process ('wabo'), coselog project," 2014. https://data.4tu.nl/repository/uuid:26aba40d-8b2d-435b-b5af-6d4bfbd7a270

[24] Mannhardt, F. (Felix), "Sepsis cases - event log," 2016. https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[25] A. Adriansyah, B. F. van Dongen, and W. M. van der Aalst, "Conformance checking using cost-based fitness analysis," in *EDOC*. IEEE, 2011, pp. 55–64.

[26] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Measuring precision of modeled behavior," *Inf. Syst. E-Business Management*, vol. 13, no. 1, pp. 37–67, 2015.

[27] S. Spiekermann and L. F. Cranor, "Engineering privacy," *IEEE Trans. Software Eng.*, vol. 35, no. 1, pp. 67–82, 2009.

[28] J. Hoepman, "Privacy design strategies - (extended abstract)," in *IFIP TC SEC*, IFIP AICT, vol. 428. Springer, 2014, pp. 446–459.

[29] W. M. P. van der Aalst, "Responsible data science: Using event data in a "people friendly" manner," in ICEIS, LNBIP, vol. 291. Springer, 2016, pp. 3–28.