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

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

[a] *Humboldt-Universität zu Berlin, Germany*
[b] *University of Mannheim, Germany*

Abstract

Information systems support the execution of business processes. As part of that, data about process execution is recorded in event logs, which can be used to analyze the control-flow of the respective processes. However, such data may contain personal information on process stakeholders that is protected by privacy regulations. Process analysis based on event logs shall, therefore, employ anonymization techniques. In this paper, we introduce two approaches to anonymize the recorded control-flow of a process. Specifically, we present SaCoFa and SaPa as two techniques to anonymize the result of trace-variant queries over an event log. Unlike existing techniques that achieve differential privacy through randomized noise insertion, our techniques rely on noise insertion mechanisms that incorporate a process' semantics, thereby avoiding easily-recognizable noise. Both techniques take different design choices, though. SaCoFa anonymizes a trace-variant distribution directly, thereby focusing on utility preservation at the expense of potentially changing the number of a traces in the result considerably. SaPa, in turn, anonymizes a trace-variant distribution indirectly, through play-out of an anonymized directly-follows distribution. This way, the number of traces in the result is close to the original log, but the drop in utility

*Corresponding author

Email addresses: `stephan.fahrenkrog-petersen@hu-berlin.de` (Stephan A. Fahrenkrog-Petersen), `martin.kabierski@hu-berlin.de` (Martin Kabierski), `han.van.der.aa@uni-mannheim.de` (Han van der Aa), `matthias.weidlich@hu-berlin.de` (Matthias Weidlich)

may become larger due to using only local control-flow information. However, our experiments demonstrate that both approaches strike a better balance of preserving the utility of an event log compared to existing techniques.
Keywords: Privacy-preserving Process Mining, Differential Privacy, Anonymization

---

1. Introduction

Information systems that support the execution of business processes often leave data traces that enable operational analysis. These data traces are sequences of events that indicate *which* activities have been conducted *when*, *by whom*, and *for whom*. Once such data is available, stored in the form of event logs, methods for process mining support a rich set of analysis questions [1]. For instance, event logs serve the construction of process models [2], the verification of the recorded executions against some specification [3], and the creation of models for simulation [4], prediction [5], or even recommendation [6].

In many application scenarios, however, event logs may contain sensitive information on process stakeholders, i.e., people that get served or are involved in the conduct of a process' activities. As such, there is an inherent risk for privacy intrusion that should be addressed for both, ethical as well as legal reasons (as imposed by, for instance, the GDPR [7] and the CCPA [8]). Yet, simply deleting or transforming identifying information is insufficient in the context of process analysis, since the behavioural characteristics of event logs pose a generally large re-identification risk [9]. That is, behavioural properties of recorded execution sequences can be linked to the context of process execution, thereby revealing the identity of process stakeholders.

Against this background, *privacy-preserving process mining* [10] strives to protect sensitive information in event logs and process mining results. To this end, an event log, or the result of some query evaluated over it, may be transformed in order to provide well-known privacy guarantees, including group-based privacy notions such as k-anonymity [11] and its derivatives [12] or

Table 1: Illustration of a trace-variant distribution, both original and privatized.

(a) Original distribution.

Trace Variant	#
$\langle Register, Triage, Surg., Release\rangle$	20
$\langle Register, Triage, Surg., Antibio., Release\rangle$	12
$\langle Register, Triage, Antibio., Antibio..Release\rangle$	6
$\langle Register, Triage, Antibio., Surg., Release\rangle$	5
$\langle Register, Triage, Consul., Release\rangle$	2
$\langle Register, Triage, Consul., Surg., Release\rangle$	4

(b) Distribution privatized using existing work.

Trace Variant	#
$\langle Register, Triage, Surg., Release\rangle$	192
$\langle Register, Triage, Antibio., Antibio., Release\rangle$	7
$\langle Release, Triage, Triage, Surg., Register\rangle$	4

differential privacy [13]. However, the transformations to achieve anonymization typically induce some loss in the utility of the data for an analysis task. As a consequence, for a given notion of utility, anonymization can typically be phrased as an optimization problem: One strives for achieving a pre-defined privacy guarantee, while minimizing the incurred loss in data utility.

One important notion of utility in process mining relates to the *trace-variant distribution* of an event log. It provides an abstract view on the control-flow of a process by listing all variations of the recorded sequences of activity executions along with their occurrence frequency. For instance, Table 1a exemplifies a trace-variant distribution for a healthcare process, which covers different clinical pathways of patients in a hospital along with their frequency. Such a trace-variant distribution provides a basis for many process mining use cases, from the exploration of the observed behaviour, through the separation of common and rare trace variants, to the construction of simulation and prediction models.

Given the importance of queries to compute a trace-variant distribution, their anonymization has previously been addressed by Mannhardt et al. [14]. Their technique uses noise insertion to achieve differential privacy, i.e., to limit the impact of one's individual data on a trace-variant distribution. Essentially, such a technique thus turns a query for a trace-variant into a probabilistic one that aims to approximate the true distribution, while satisfying a desired privacy guarantee. While this state-of-the-art technique succeeds in its privatization purpose, the work, however, neglects three important aspects:

(1) Anonymized distributions may include trace variants that denote behaviour that was never observed or, more importantly, which is clearly impossible for the considered process. Even if the distribution is generally close to the one over the original log, ignoring a process' semantics means that inserted noise may be identified immediately. For instance, after privatizing the event log in Table 1a, naive noise insertion may yield variants like the last one in Table 1b, in which patients are released before being treated in a hospital, whereas such behaviour is naturally never seen in reality.

(2) A privatized distribution may show fundamentally different absolute properties compared to the true distribution. In particular, the number of traces in the result tends to deviate by up to orders of magnitude, which lowers its utility and clearly indicates that the data has been transformed. For instance, the original distribution in Table 1 indicates that 49 traces were executed in the process, whereas the privatized distribution clearly overshoots this number, with a total of 203 traces in the distribution, 192 of which correspond to its most common variant.

(3) Finally, the state of the art requires the selection of parameters that have a large influence on the obtained result (e.g., the length of trace prefixes to prune [14]), but for which hardly any guidance can be given for a specific application scenario. This generally lowers the practical applicability of the technique and may have far reaching consequences for the results. For instance, depending on the setting for such a pruning parameter, entire swaths of behaviour may be omitted from the privatized trace-variant distribution. This is, e.g., seen for the variants starting with $\langle \mathit{Register}, \mathit{Triage}, \mathit{Consul.} \rangle$, which were part of the original event log, but no longer appear in Table 1b.

In previous work [15], we addressed the first of the above issues with SaCoFa, an approach for semantics-aware control-flow anonymization. Unlike existing techniques that achieve differential privacy by the Laplace mechanism, which inserts noise randomly, SaCoFa employs the exponential mechanism, which enables the incorporation of the semantics of the underlying process. By nudging noise insertion towards trace variants satisfying certain behavioural constraints, we achieve

trace variant distributions that show less unobserved or impossible behaviour, while providing the same privacy guarantee. However, SaCoFa still follows the same idea as existing techniques to construct trace variants, i.e., it iteratively builds up a prefix tree. Therefore, it inherits the aforementioned limitations related to the absolute properties (2) and the practical applicability (3).

In this paper, we close this research gap and complement SaCoFa with an alternative algorithm to construct differentially-private trace variant distributions. To this end, we present a step-wise construction of trace variants using an anonymized version of the distribution of the directly-follows relation of an event log, referred to as semantics-aware play-out-based anonymization, SaPa in short. This way, we better preserve the number of traces, compared to SaCoFa. At the same time, SaPa does not necessitate a user to specify algorithmic parameters, which, due to the lack of an intuitive interpretation, are hard to determine. By integrating the ideas of SaCoFa on nudging privatization towards trace variants that satisfy behavioural constraints in SaPa, we also achieve semantics-aware anonymization.

While the newly presented algorithm addresses all three of the above issues, it also comes with drawbacks compared to SaCoFa. In particular, the utility in terms of the relative similarity of the resulting distribution and the true distribution may be worse for relatively structured processes. This is due to SaPa exploiting only local control-flow information (if executions of two activities may directly follow each other), whereas SaCoFa always considers complete prefixes of traces.

As a consequence, both algorithms open a design space for semantics-aware anonymization of trace-variant distributions. We explore this design space in comprehensive evaluation experiments with several real-world event logs. Our empirical results illustrate that both algorithms yield higher utility for process analysis and provide more robust privacy guarantees than the state of the art. Moreover, they introduce less obvious noise in the result, as classified by anomaly detection techniques. Comparing the two algorithms, we note that SaPa properly maintains the number of traces in all evaluation scenarios. In terms of the utility

of the trace-variant distribution, both algorithms are mostly on-par for relatively unstructured processes, while SaCoFa leads to better utility for scenarios with highly structured behaviour.

In the remainder, Section 2 provides essential background information and definitions. Our two algorithms, SaCoFa and SaPa are introduced in Section 3 and Section 4, respectively. Our evaluation results are summarized in Section 5, whereas key insights are discussed in Section 6. Finally, we review related work in Section 7 and conclude in Section 8.

2. Background

Event model. Our work focuses on the control-flow perspective of business processes. Therefore, we use an event model that builds upon a universe of activities $\mathcal{A}$. Each event in a log is assumed to correspond to one of these activities. Using $\mathcal{E}$ to denote the universe of all events, a single execution of a process, i.e., a *trace*, is modelled as a sequence of events $t = \langle e_1, e_2, \ldots, e_n \rangle \in \mathcal{E}^*$, such that no event can occur in more than one trace. We write t_i, $1 \leq i \leq n$, to refer to the i-th event e_i of the trace t. An event log is a set of traces, $L \subseteq 2^{\mathcal{E}^*}$, with $\mathcal{L}$ as the universe of event logs. Distinct traces that indicate the same sequence of activity executions are said to be of the same *trace variant*, i.e., $\mathcal{A}^*$ is the universe of trace variants. The set of activities referenced by events in an event log L is denoted by $\mathcal{A}(L)$.

Trace-variant queries. A *trace-variant query* is a function that returns the trace-variant distribution of an event log L, i.e., it captures how often certain trace variants occur in L. With $\mathcal{A}^*$ as the universe of potential trace-variants and $\mathbb{N}$ is the set of natural numbers, it is defined as follows:

$$\tau(L) : \mathcal{L} \rightarrow \mathcal{A}^* \times \mathbb{N}.$$

In other words a trace-variant query $\tau(L)$ returns a count for each trace-variant that exists in the log. Aside from $\tau(L)$ we also define a query that returns the

number of times a trace variant v occurs in L:

$$\tau(L, v) : \mathcal{L} \times \mathcal{A}^* \rightarrow \mathbb{N}.$$

Directly-follows queries. A *directly-follows query* is a function $\delta(L)$ that returns the directly-follows distribution of an event log L, i.e., it captures how often an event e_1 corresponding to an activity $a_1 \in \mathcal{A}$ is directly-followed by an event e_2 corresponding to an activity $a_2 \in \mathcal{A}$ within the same trace t:

$$\delta(L) : \mathcal{L} \rightarrow \mathcal{A} \times \mathcal{A} \times \mathbb{N}.$$

A directly-follows query $\delta(L)$ is a special case of a trace-variant query $\tau(L)$, in the sense that a trace-variant query is counting sequences of varying length in a log L and a directly-follows-query $\delta(L)$ counts sequences of fixed length two (reflected by the domain $\mathcal{A} \times \mathcal{A}$), i.e., pairs of directly-following activities. Therefore, the following discussion on differential privacy and insights on how it relates to trace-variant queries τ, also implicitly applies to directly-follows queries δ. We therefore focus on the trace-variant-query in the remainder of this section and limit the discussion to hints on how the respective concepts are adopted to a directly-follows query.

Differential privacy. A query can be made privacy-preserving, through a privacy guarantee [16], such as *differential privacy* [13], which has been adopted by companies such as Apple [17], SAP [18], and Google [19]. The main benefit of differential privacy is its immunity to post-processing. Therefore, it provides an effective protection and limits the information gain of an adversary. The general idea behind differential privacy is to ensure that the inclusion of the data of one individual in a certain dataset does not significantly change the result returned by a query over this data. In the context of our work, this implies that a trace-variant query τ is said to preserve differential privacy, if the trace-variant distribution returned by query $\tau(L)$ does not significantly differ from the distribution returned by a query over a *neighbouring* log, i.e., a log that contains one additional trace, $\tau(L \cup \{t\})$ or misses a certain trace $\tau(L \setminus \{t\})$, for any trace $t \in \mathcal{E}^*$. Note that, for a directly-follows query, *neighbouring* logs would

actually be characterized by the presence of a single pair of events that affects the directly-follows relation for one pair of activities. However, we can adopt the above definition based on the difference by a single trace in the absence of repetitive structures in the traces, since anonymization of different pairs of activities is independent of each other (known as the parallel composition rule of differential privacy [20]).

A trace-variant query τ that returns the actual frequency distribution, in general, cannot be expected to satisfy differential privacy. Hence, one relies on probabilistic queries $\hat{\tau}$ that approximate the true distribution, while satisfying the privacy guarantee. This leads to the following definition:

Definition 1 (Differential Privacy). *Given a probabilistic trace-variant query $\hat{\tau}$ and privacy parameter $\epsilon \in \mathbb{R}$, query $\hat{\tau}$ provides* ϵ-differential privacy*, if for all neighbouring pairs of event logs $L_1, L_2 \in \mathcal{L}$ and for all sets of possible trace-variant distributions, $D \subseteq \mathcal{A}^* \times \mathbb{N}$, it holds that:*

$$Pr[\hat{\tau}(L_1) \in D] \leq e^{\epsilon} \cdot Pr[\hat{\tau}(L_2) \in D]$$

where the probability is taken over the randomness introduced by the query $\hat{\tau}$.

The lower the value of ϵ, the stronger the provided privacy guarantee. In scenarios where an individual can be part of the result multiple times (i.e., in multiple traces for a trace-variant query or through multiple repetitions of pairs of activities for a directly-follows query), the same degree of privacy can be achieved by dividing the privacy parameter ϵ by the maximal number of occurrences of an individual in the result. In case of the directly-follows query such a lower ϵ could only be applied to the respective directly-follows relation.

To establish a trace-variant query $\hat{\tau}$ that provides ϵ-differential privacy as given in Def. 1, it is common to insert noise into the result of the original query τ, thus obfuscating the absolute occurrence counts of potential trace variants. This is commonly done according to a noise-insertion mechanism, guided by a certain probability distribution. In this work, we consider the Laplace and exponential mechanisms for this purpose, as described in the following paragraphs.

Laplace mechanism. The Laplace mechanism inserts noise based on a Laplace distribution and was previously used to anonymize trace-variant distributions [14]. The impact of this mechanism generally depends on the strength of the privacy guarantee ϵ and the *sensitivity* Δf of some query q. A query $\hat{q}$ protected by the Laplace mechanism can formally be described as:

$$\hat{q} \leftarrow q + Lap(\frac{\Delta f}{\epsilon})$$

The sensitivity Δf depends on the maximum impact one individual can have on the result of query q. So, if q is a trace-variant query (τ, as introduced above) and one individual participates in at most one trace, the sensitivity is $\Delta f = 1$. In the absence of repetitive structures in the traces, the same holds true for the directly-follows query, as, again, the anonymization of different pairs of activities is independent of each other. If an individual can appear multiple times in the query result, the sensitivity is higher and more noise needs to be introduced to achieve ϵ-differential privacy. However, in such scenarios, the guarantee of ϵ-differential privacy may also be relaxed, which lowers the increase in sensitivity and still provides a relatively strong protection [21].

When used to insert noise into a trace-variant distribution, the Laplace mechanism has considerable drawbacks. That is, the probability for a certain anonymized trace-variant distribution to be returned only depends on the syntactic distance of this distribution to the actual one. Yet, this ignores that certain distributions are less desirable than others, even when they are syntactically just as different.

Exponential mechanism. The exponential mechanism [22] enables prioritization of certain query results by incorporating the notion of a score function into the noise-insertion process. This score function $s : \mathcal{L}, D \rightarrow \mathbb{R}$ defines some results to be more desirable than others for the given dataset over which the query is evaluated. Put differently, the higher $s(d, r)$, the more desirable query result r is for the dataset d. Moreover, let Δs be the sensitivity of score function s, i.e., the maximum differences between scores assigned to the possible results for any two neighbouring datasets. Then, for some query q over dataset d and

privacy parameter ϵ, the query $\hat{q}$ protected by the exponential mechanism [22] is derived by selecting result r with a probability proportional to $e^{(\epsilon s(d,r))/(2\Delta s)}$.

The exponential mechanism may be lifted to trace-variant queries, as follows: For a given log L and a possible trace-variant distribution $D \in \mathcal{A}^* \times \mathbb{N}$, the score $s(L, D)$ shall capture whether D is desirable in terms of a process' semantics, as captured by L. Then, the mechanism returns a specific trace-variant distribution D with a probability proportional to $e^{(\epsilon s(L,D))/(2\Delta s)}$.

This general idea will be exploited in our algorithms, SaCoFa and SaPa, as presented in the next sections.

3. Semantics-aware Control-flow Anonymization

This section introduces SaCoFa (semantics-aware control-flow anonymization) as an approach to retrieve the anonymized behaviour of an event log. Section 3.1 presents the general algorithm based on the exponential mechanism. Section 3.2 then defines the score function that SaCoFa employs for incorporating a process' semantics. Finally, Section 3.3 discusses pruning strategies for SaCoFa, their computational necessity, and how the score function helps to decrease the negative effects of pruning.

3.1. The SaCoFa Algorithm

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

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

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

First, the algorithm initializes the prefix tree, represented as an empty set of prefixes T (line 1). Next, the trace-variant distribution d and the current prefix length n are initialized (lines 2-3). Then, the prefix tree is iteratively expanded. This expansion will eventually terminate once n reaches the maximal prefix length k (line 4).

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

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

Algorithm 1: The SaCoFa Algorithm

input : L, an event log; ϵ, the privacy parameter; k, the max. prefix length; p ($p_{harmless}, p_{harmful}$), the pruning parameter(s).

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

1 $T \leftarrow \{\langle\rangle\}$; /* Initalize the prefix tree with an empty prefix */

2 $d \leftarrow \emptyset$; /* Initalize the trace-variant distribution */

3 $n \leftarrow 1$; /* Initialize the current prefix length */

4 **while** $n \leq k$ **do** /* Consider prefixes up to length k */

5 $\quad C \leftarrow \emptyset$; /* Initialize candidate set */

$\quad$ /* Select candidate prefixes to expand */

6 $\quad$ **foreach** $v \in T \wedge |v| = n-1 \ \wedge v(|v|) \neq \perp$ **do**

$\quad\quad$ /* For each possible activity */

7 $\quad\quad$ **foreach** $a \in \mathcal{A}(L) \cup \{\perp\}$ **do**

$\quad\quad\quad$ /* Add expanded prefix to candidate set */

8 $\quad\quad\quad C \leftarrow C \cup \{v.\langle a\rangle\}$;

$\quad$ /* Determine harmless prefix candidates */

9 $\quad C_{expand} \leftarrow \{c \in C \mid \mathbf{score}(L, c) = 1\}$;

$\quad$ /* Determine harmfull prefix candidates */

10 $\quad C_{harm} \leftarrow C \setminus C_{expand}$;

$\quad$ /* Select prefixes; harmful prefix candidates are selected using the exponential mechanism */

11 $\quad C_{expand} \leftarrow C_{expand} \cup Exp(C_{harm}, \mathbf{score}(L, .), \epsilon)$;

$\quad$ /* Assign positive noisy count to prefixes */

12 $\quad$ **foreach** $v \in C_{expand}$ **do**

13 $\quad\quad d(v) \leftarrow [\tau(L, v) + Lap(\frac{1}{\epsilon})]_{>0}$;

14 $\quad T \leftarrow \mathbf{prune}(T, d, p, C_{harm})$; /* Prune prefix tree */

15 $\quad n \leftarrow n + 1$; /* Increase current prefix length */

/* Return the distribution over all prefixes that are complete or of length k */

16 **return** $\{d(v) \mid v \in T \wedge (v(|v|) = \perp \vee |v| = k)\}$;

Tree expansion. The candidate prefixes in C are evaluated with a *score function* to classify them as harmless (C_{expand}) or harmful (C_{harm}) (lines 9-10). The definition of the score function depends on the incorporated notion of a process' semantics and will be discussed in Section 3.2. Here, we assume the

score function to be applicable to prefixes, while the actual scoring (function s in Section 2) refers to a distribution over prefixes with their frequencies all set to one.

Employing the exponential mechanism, we determine which of the harmful prefixes to add to the tree by random selection (line 11). Then, the selected harmful prefixes, together with the harmless ones, expand the prefix tree (line 12). Each of these prefixes is assigned a noisy count, based on its number of occurrences in the original log and added random noise (line 13). The Laplace noise is drawn from a Laplace distribution, scaled according to the privacy parameter ϵ, i.e., $Lap(\frac{1}{\epsilon})$ (as discussed in Section 2). Here, we enforce that the noisy count is positive, since the decision to include the prefix has already been taken as part of the exponential mechanism.

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

Tree pruning. Following the prefix tree expansion, we prune it based on the noisy counts assigned to trace variants (line 14). A simple pruning strategy removes all prefixes from the tree, for which the noisy count is below a threshold set by parameter p. However, as we will discuss in Section 3.3, pruning may treat harmful and harmless prefixes differently (using two thresholds, $p_{harmless}$ and $p_{harmful}$). In general, we also favour pruning of harmful prefixes to avoid the removal of prefixes that conform to the semantics of the process at hand.

Let us revisit our example and let us assume the pruning parameter was set to $p = 5$. We also assume no difference is made between harmful and harmless prefixes. In this case, the noisy count for the prefix $\langle Register, Triage \rangle$ with 44 is higher than the pruning parameter and the prefix is kept inside the prefix tree T. Therefore, this prefix will be considered within the further extension of the prefix tree. On the other hand, for the prefix $\langle Register, Register \rangle$, we have a

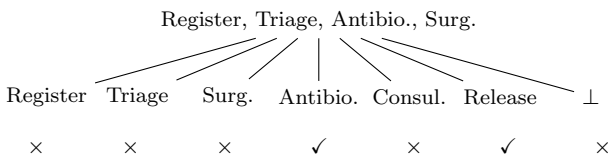


Figure 1: Example of potential prefix extensions. ✓ and ×, indicate harmless and harmful prefix extensions, respectively.

noisy count of 3 and the prefix is removed from the prefix tree. Prefixes that would be based on $\langle \mathit{Register}, \mathit{Register} \rangle$ will also no longer be considered in the remaining run of SaCoFa.

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

3.2. A Semantics-aware Score Function

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

To exemplify this design choice, we propose a function that is based on a generalization of the behaviour in the original log. Specifically, we consider a behavioural abstraction that was proposed in the context of the *behavioural appropriateness* measure [23]. This behavioural abstraction defines rules between pairs of activities, reflecting their order and co-occurrence in a log. Specifically, given $a_1, a_2 \in \mathcal{A}(L)$, the rules capture if a_1 will *always*, *never*, or *sometimes* follow (or precede) an activity a_2, not necessarily directly. As such, the set of rules encodes hidden business logic, derived from a log without manual intervention.

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

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

For illustration purposes, consider the example given in Fig. 1, which depicts the expansion of prefix $\langle \mathit{Register}, \mathit{Triage}, \mathit{Antibio.}, \mathit{Surg.} \rangle$, based on the example from Table 1. In the original log, activity *Surg.* is always followed by activity *Release*, may be followed by activity *Antibio.*, and is never followed by the remaining activities. Respecting these behavioural rules, expansions based on the former two activities are considered harmless, while those in the latter are categorized as harmful.

As mentioned above, the score function may also be defined based on other behavioural models. In particular, it may be grounded in other sets of behavioural rules, such as those presented in [24, 25], which are then instantiated for the original log to capture the semantics of the underlying process. Moreover, rules may also originate from other sources, such as textual documents [26]. However, deriving the rules from the original log ensures that trace variants in the original log are more likely to be preserved.

3.3. Semantics-aware Pruning

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

The need for generalization. Pruning comes with the risk of removing prefixes (and thus trace variants) that are common in the original log, which subsequently reduces the utility of the anonymized trace-variant distribution. However, unlike log anonymization with the Laplace mechanism, SaCoFa supports a differentiation between prefixes that are harmful and harmless for an anonymized distribution. Therefore, we can limit pruning only to harmful pre-

fixes. This way, the overall number of pruned prefixes is reduced, but harmless prefixes are always preserved, even when their noisy count is below the pruning parameter p.

A lower number of pruned prefixes, in general, also reduces privacy degradation. However, when pruning solely harmful prefixes, there is a risk to violate the required differential privacy guarantee. That is, if harmful prefixes are characterized based on their absence in the original log, the following may happen: For two neighbouring event logs, that differ by a trace of a variant that appears only in one of the logs, the anonymized variant distributions may enable the identification of the respective trace. To avoid such situations, we employ a pruning strategy that incorporates behavioural generalization.

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

For illustration, consider a log L_1 containing only traces that represent variants from Table 1a. Let $L_2 = L_1 \cup \{t\}$ be a neighbouring log, where t is a trace of the variant $\langle \mathit{Register}, \mathit{Antibio.}, \mathit{Release} \rangle$. Comparing the rule sets of both logs, trace t adds the rule that *Register* is sometimes followed by *Antibio*. Hence, the SaCoFa algorithm would consider the prefix $\langle \mathit{Register}, \mathit{Antibio.} \rangle$ as harmless when anonymizing L_2, whereas it would be harmful regarding L_1. For L_2, further prefixes would then be derived and considered as harmless, e.g., $\langle \mathit{Register}, \mathit{Antibio.}, \mathit{Release} \rangle$ and $\langle \mathit{Register}, \mathit{Antibio.}, \mathit{Surg.}, \mathit{Release} \rangle$. Thus, the distributions derived for the logs will differ by more than one trace variant.

Therefore, pruning only harmful prefixes requires that a single trace either leads to multiple trace variants to be considered as harmless, or none at all. In

practice, this may not be the case, which is why we relax the pruning strategy, as follows. We introduce $p_{harmless}$ and $p_{harmful}$ as separate pruning thresholds for harmless and harmful prefixes, respectively. By setting $1 < p_{harmless} < p_{harmful}$, we favour pruning of harmful prefixes. Yet, by pruning also some harmless prefixes, we ensure that information on the existence of a single trace variant is not disclosed, even if the above requirement is not met. Also, the two aforementioned extreme scenarios could be configured accordingly, i.e., pruning only harmful traces ($p_{harmless} = 1$ and $p_{harmful} > 1$) or pruning all prefixes that introduce new behaviour ($p_{harmless} = 1$ and $p_{harmful} = \infty$). It is important to note that pruning is always applied to the noisy counts, as inserted during the *Tree expansion* step.

4. Directly-follows-based Control-flow Anonymization

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

Our idea is to construct a trace-variant distribution based on a directly-follows query and a play-out procedure for the obtained result. In Section 4.1, we first outline the general idea behind SaPa and explain why this approach avoids the aforementioned issues. Afterwards, in Section 4.2, we lift the idea of semantics-aware anonymization to the anonymization of the directly-follows query, allowing us to instantiate SaPa in a semantics-aware manner as well.

4.1. Generating Trace Variants based on Play-out

A directly-follows distribution, capturing how often events corresponding to certain activity pairs directly follow each other in traces (see Section 2), denotes

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

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

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

Anonymization of the directly-follows distribution. The first step in the approach is to anonymize the directly-follows distribution. Our algorithm is independent of the choice for a specific procedure to achieve this.

As suggested in literature, a common choice would be a procedure based

Algorithm 2: Play-out Algorithm

input : L, an event log.

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

1 $d \leftarrow \emptyset$; /* Initialize the trace-variant distribution */

2 $dfg' \leftarrow \hat{\delta}(L)$; /* Generate ϵ-differentially private directly-follows distribution */

3 **while** $containsTrace(dfg')$ **do** /* Checks if there is still a trace */

4 $\quad t \leftarrow \langle \top \rangle$; /* Initalize trace with trace start event */

$\quad$ /* As long as last element is not end of trace and it is not empty */

5 $\quad$ **while** $t_{|t|} \neq \bot \vee t \neq \langle\rangle$ **do**

$\quad\quad$ /* Select next activity based on random choice from the df-distribution */

6 $\quad\quad a \leftarrow pickNextActivity(dfg', t_{|t|})$;

$\quad\quad$ /* Check if a next activity exists and the trace can be continued */

7 $\quad\quad$ **if** $a \neq \varnothing$ **then**

$\quad\quad\quad$ /* Decrease DFG count by used directly-follows relation */

8 $\quad\quad\quad dfg' \leftarrow dfg' \setminus \{(t_{|t|}, a, n)\} \cup \{(t_{|t|}, a, n-1)\}$;

9 $\quad\quad\quad t \leftarrow t.\langle a \rangle$; /* Adding a to t to continue the trace */

10 $\quad\quad$ **else**

$\quad\quad\quad$ /* All df-relation entries to the last element of t are removed */

11 $\quad\quad\quad dfg' \leftarrow \{(a_1, a_2, n) \in dfg' \mid a_2 \neq t_{|t|}\}$

12 $\quad\quad\quad t \leftarrow \langle t_1, ..., t_{|t|-1} \rangle$; /* The last element of t is removed */

$\quad$ /* Trace variant distribution is updated by including the current trace */

13 $\quad$ **if** $t \neq \langle\rangle$ **then** $d(t) \leftarrow \begin{cases} d(t) + 1 & \text{if } t \in \text{dom}(d) \\ 1 & \text{otherwise.} \end{cases}$;

/* Return anonymized trace variant distribution */

14 **return** d

on the Laplace mechanism [14], configured by the privacy parameter ϵ. We later propose an alternative procedure that accounts for a process' semantics (Section 4.2). In any case, it is important to note that, contrary to the approaches that anonymize trace-variants directly, we here anonymize each entry of the directly-follows relation independently.

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

We provide an example of an anonymized directly-follows distribution in Table 2. The example serves illustrative purposes and is, therefore, less noisy than what would be expected to increase readability.

Play-out of the anonymized directly-follows distribution. Next, traces will be generated from the directly-follows distribution, for as long as a trace can be constructed from the start symbol ($\top$) to the end symbol ($\bot$) based on the entries in the directly-follows distribution (line 3). Here, the respective entries in the directly-follows distribution need to be subsequent, i.e., the target activity of one entry needs to be equivalent to the source of the next entry, and their counts in the distribution need to be larger than zero. If such a construction would be generally possible, beginning with a start symbol ($\top$) (line 4), we try to assemble a trace activity by activity until reaching the end symbol ($\bot$) (line 5). Once this happens, the now completed trace is included in the trace-variant distribution (line 13).

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

Let us turn to our example in Table 2 to better understand these steps. In the example, a trace could be started with either the activity *Register* or *Triage*. Assume that we start a trace we *Triage*. As the next step, we could extend the the trace with either *Surg.* or *Antibio.*. If the trace is extended with *Surg.*, it is only possible to continue to *Release*. From there, we finally reach the trace end $\bot$ and we would have constructed a trace $\langle Triage, Surg.Release \rangle$. We now would decrease all counts that let to that trace and restart with $\top$.

If no activity could be found for expansion of the trace, we remove all entries of the directly-follows distribution that lead to this activity (line 11). Furthermore, aiming for completion of the current prefix, we adopt a backtracking procedure.

Table 2: Example of a anonymized Directly-follows Distributions

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

That is, we remove the last activity (line 12), and continue with the algorithm. Should it turn out to be impossible to finish construction of the respective trace, backtracking will yield the empty trace, which is discarded entirely.

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

Eventually, it will no longer be possible to construct any traces, from start to end, based on the remaining counts of the directly-follows distribution. In that case, the trace-variant distribution is returned (line 14).

4.2. Exponential Mechanism for Directly-follows-Query

In this section, we propose a semantics-aware procedure for the anonymization of a directly-follows query, based on the exponential mechanism. The exponential mechanism aims to optimize the model generated by the play-out of the resulting directly-follows distribution. Therefore, the main goals are to prevent the insertion of harmful directly-follows relations and to preserve relations that

provide utility. Our intuition to achieve that goal is based on the idea, that we want to avoid the procedure skipping parts of the process, by jumping from the start of the process to its end.

To ensure this intuition, we assess the utility of a directly-follows relation, i.e., how realistic its occurrence is, in terms of its *k-follows score*. This score measures the minimum distance that exists between pairs of activities, for the traces in an event log. For instance, if, following events corresponding to an activity a_1, there are always at least two other events before an event of activity a_2 appears, the k-follows score from a_1 to a_2 is 3.

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

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

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

Exponential mechanism for the directly-follows query. The algorithm takes as input an event log L, a parameter k for the score function and a parameter ϵ to determine the privacy protection guarantee. First, the directly-follows distribution (line 1) is initialized. Next, all directly-follows relations of the original event log L are separated into utility-preserving R_{util} (line 2 and harmful R_{harm}(line 3), depending if their k-follows score is below or equal to, or

Algorithm 3: Exponential Mechanism for the Directly-follows Query

input : L, an event log, ϵ, the privacy parameter; k, the k-follows threshold.

output : dfg', an anonymized directly-follows distribution.

1 $dfg' \leftarrow \emptyset$

2 $R_{util} \leftarrow \{(a_1, a_2, n) \in \delta(L) \mid \textbf{follows}-\textbf{score}(a_1, a_2) \leq k\}$

3 $R_{harm} \leftarrow \{(a_1, a_2, n) \in \delta(L) \mid \textbf{follows}-\textbf{score}(a_1, a_2) > k\}$

4 $R_{util} \leftarrow R_{util} \cup Exp(R_{harm}, \textbf{score}(L, .), \epsilon)$

5 **foreach** $(a_1, a_2, n) \in \delta(L)$ **do**

6   **if** $(a_1, a_2, n) \in R_{util}$ **then**

7         $n' \leftarrow [n + Lap(\frac{1}{\epsilon})]_{>0}$

8     $dfg' \leftarrow dfg' \cup \{(a_1, a_2, n')\}$

9 **return** dfg'

above the threshold k.

Now, some randomly picked harmful relations from R_{harm} are added to those relations that will be preserved R_{util}, whose amount depends on the privacy parameter ϵ (line 4). After this, the procedure iterates through the directly-follows distribution (line 5) and all directly-follows relations captured in R_{util} are considered for further anonymization (line 6). Noise scaled proportionally to the privacy guarantee ϵ is added to the considered relations. At the same time, we enforce that these relations need to have counts above 0, so they need to be present in the anonymized directly-follows query result (line 7). The anonymized directly-follows distribution is updated with the new privatized occurence count, and the procedure continues with the next relation (line 8). Once all directly-follows relations have been considered, the anonymized directly-follows distribution is provided as an output (line 9).

5. Evaluation

In this section, we evaluate our approaches to control-flow anonymization with SaCoFa and SaPa, which includes a comparison against the state of the art. We investigate the utility of the derived trace-variant distributions for process

discovery and assess the ability to avoid obvious noise by exploring the frequency of anomalies.

Below, we first review the used datasets (Section 5.1) and our experimental setup (Section 5.2). We then present our experimental results (Section 5.3).

5.1. Datasets

We use three real-world event logs as the basis for our experiments. Table 3 lists some essential properties of these event logs, illustrating also the motivation for selecting them: The event logs differ in their size and complexity. The *Traffic Fines* log contains data on a very structured process, with just 231 variants over a total of 150,370 traces. In contrast, the *Sepsis* log captures an unstructured hospital-treatment process, containing 846 variants of which the vast majority occurred just once. Finally, the *CoSeLoG* log provides a middle ground, with a semi-structured process that consists of 116 variants over 1,434 traces.

Table 3: Descriptive statistics for the event logs.

Event Log	# Events	# Activities	# Cases	# Variants
CoSeLoG [27]	8,577	27	1,434	116
Sepsis [28]	15,214	16	1,050	846
Traffic Fines [29]	561,470	11	150,370	231

5.2. Experimental Setup

Baselines. We evaluate our approaches against the state of the art for the computation of anonymized trace-variant distributions, as presented by Mannhardt et al. [14]. It anonymizes the result of a trace-variant query using the Laplace mechanism, which is why we refer to this approach as ‘Laplace‘. Furthermore, we also consider a realization of SaPa using a directly-follows distribution that was anonymized with the Laplace mechanism, as also suggested by Mannhardt et al. [14]. We refer to this approach as ‘DF-Laplace’.

Parameter settings. SaCoFa takes four parameters: the strength of the desired privacy guarantee ϵ, an upper bound on the trace-variant length k, and the

Table 4: Employed parameter settings.

Log	ϵ	k	$p_{harmful}$	$p_{harmless}$
	1.0	10	3	/
CoSeLoG	0.1	10	25	22
	0.01	10	220	200
	1.0	23	4	/
Sepsis	0.1	23	20	15
	0.01	23	190	150
	1.0	9	2	/
Traffic Fines	0.1	9	20	15
	0.01	9	150	120

pruning parameters $p_{harmful}$ and $p_{harmless}$. Per event log, we set k so that roughly 80-90% of the original trace variants are covered. For each of the employed privacy guarantees, i.e., $\epsilon = \{1.0, 0.1, 0.01\}$, we explored pruning parameters starting at 2, 20, and 200, respectively, until a configuration was found such that the trace-variant query could be executed within several seconds. Overall, this approach resulted in the parameter settings given in Table 4, which we employed for our approach and, if applicable, for the baseline techniques.

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

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

To discover a model, we use the Inductive Miner Infrequent [30], a state-of-the-art technique for process discovery, which is commonly used (and selected here) because it guarantees process models that are free of deadlocks. The technique uses noise filtering to remove infrequent behaviour, for which we

select the default threshold of 20%. Given such a discovered model, we use several metrics to measure its utility. First, we use the F-score in relation to the original event log, i.e., the harmonic mean of the *fitness* [31] and *precision* [32]. These metrics are the most widely-used metrics for process model utility. Also, we evaluate the *generalization* [33] of the process models discovered from the anonymized trace-variant distributions. For these measures, a score closer to 1 indicates a better result. Finally, as an additional proxy for utility, we use sequence entropy [34] to measure the complexity of the anonymized logs. It was shown, that a high level of complexity leads to worse process discovery results. Therefore, a low complexity score is desirable. The complexity score is normalized between 0 and 1, with 1 being the highest level of complexity.

Beyond assessing the utility of discovered process models, we assess the size of the result by considering the ratio of the number of traces within the anonymized result and the number of traces in the original log. Therefore, a ratio above one means that the result size increased through anonymization, whereas a value below one hints at a reduction in the number of traces.

Finally, as an additional evaluation dimension, we measure the fraction of easily-recognizable noise introduced as part of the anonymization. To this end, we apply a standard anomaly detection technique, which employs isolation forests [35], to the anonymized result. We train the model on the original log, before using it to detect anomalous traces in the anonymized result. As features in the learning process, we use a binary encoding of the activities, signalling if they are present in a trace. Moreover, we also encode the presence of directly-follows dependencies in a trace with a binary encoding.

Implementation. To conduct our experiments, we implemented SaCoFa and SaPa in Python. The source code is available on GitHub[1] under the MIT license. Furthermore, we used PM4Py's [36] implementation of the Inductive Miner and the evaluation measures. The implementation of the isolation forest is available

[1] `https://github.com/samadeusfp/SaCoFa`

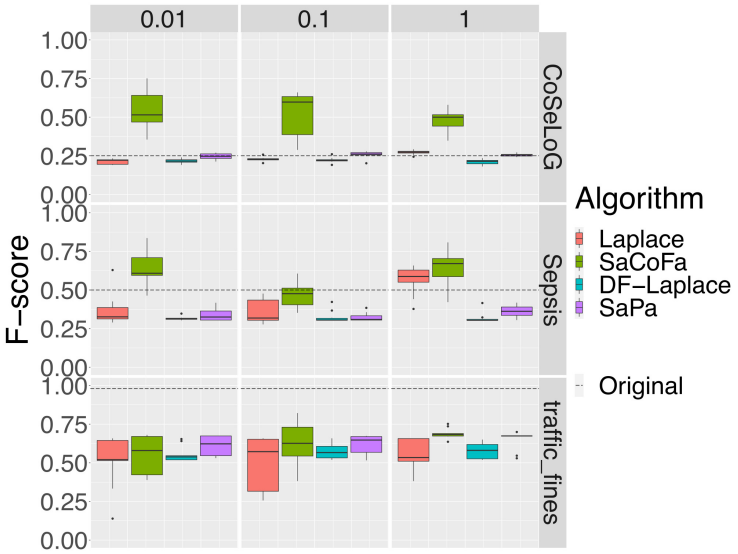


Figure 2: F-score of discovered process models.

in scikit-learn.[2]

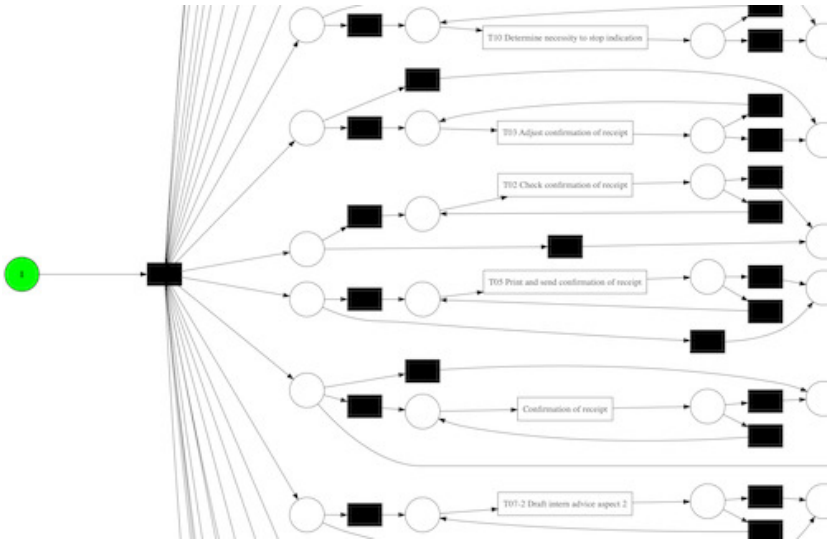
Repetitions. To account for the non-deterministic nature of the algorithms, we perform 10 repetitions of all experiments. In the remainder, we report on the median, the upper quartile, and the lower quartile, using box plots.

5.3. Results

Process Discovery Utility. Fig. 2 depicts the F-scores of the process models constructed from the anonymized trace-variant distributions derived by the different techniques. As shown, SaCoFa outperforms the other techniques considerably, being on par only for the setting with the strongest privacy guarantee ($\epsilon = 0.01$) and the most structured process (Traffic Fines).

In particular, we observe major improvements for the two less-structured processes, which have more activities and longer traces, resulting in significantly higher F-scores obtained using SaCoFa, while providing the same privacy guarantee as the other techniques. Furthermore, our results also illustrate that,

[2] `https://scikit-learn.org/stable/`



(a) Laplace baseline.

(b) SaCoFa approach.

Figure 3: Process models obtained for anonymized versions of CoSeLoG ($\epsilon = 0.01$).

sometimes, the anonymized results lead to higher F-scores than the original log. The reason being that the Inductive Miner guarantees the generation of a fitting model, which may result in very low precision values. If an anonymized log contains less behaviour, above the threshold adopted by the discovery algorithm to filter noise, the model becomes more compact. It then shows higher precision and, therefore, also a higher F-score. We also observe, that SaPa yields slightly better results than the DF-Laplace mechanism. However, both algorithms based on directly-follows queries achieve results similar to the direct anonymization of the trace-variant query with the Laplace mechanism and are therefore, no match with the results produced by SaCoFa.

To further illustrate the above results, Fig. 3 shows excerpts of two process models obtained for the CoSeLoG process under the strongest privacy guarantee ($\epsilon = 0.01$). Here, Fig. 3a shows part of the model discovered from the result of

the Laplace baseline, whereas Fig. 3b is based on SaCoFa. As seen, the process model generated with SaCoFa is much more structured. It starts with a sequence of activities that, notably, is also the same in the process model generated from the original event log. In contrast, the model in Fig. 3a is highly unstructured and strays far from the original process: nearly all activities can start a trace, be skipped, or executed multiple times. Therefore, the higher utility of the process model also comes with a better understandability of the model. This result further supports the argument that SaCoFa provides the highest utility for process discovery.

Next, we turn to an assessment of the generalization of the obtained models. As illustrated in Fig. 4, the models generated based on the results derived with SaCoFa are more general than the models generated by the Laplace baseline technique, i.e., they abstract more from the represented behaviour. SaCoFa also produces more general models than the two techniques based on the directly-follows queries, with the exception of the Traffic Fines log, i.e., the most structured event log. Combined with the results for the F-score, we conclude that the results anonymized with SaCoFa have the highest utility for process discovery. Results generated with SaPa are usually of similar quality as those generated by the baseline techniques.

As a final measurement for process discovery utility we turn to the complexity of the underlying log, measured by sequence entropy. As shown in Fig. 5, for two out of the three logs, the trace-variant-based approaches lead to less complex logs, with SaCoFa outperforming the Laplace mechanism. SaPa seems to perform slightly better than the Laplace counterpart. Overall, the biggest log (Traffic Fines) is mostly resistant to anonymization in terms of complexity. For the other logs, SaCoFa achieves lower complexity than the original logs. We attribute this to the prefix-tree pruning, which filters a certain amount of noise.

Result Size. Next, we turn to the number of traces in the trace-variant distributions generated by the respective techniques. In Fig. 6, we show the relative number of traces within the anonymized results generated by SaCoFa, SaPa,

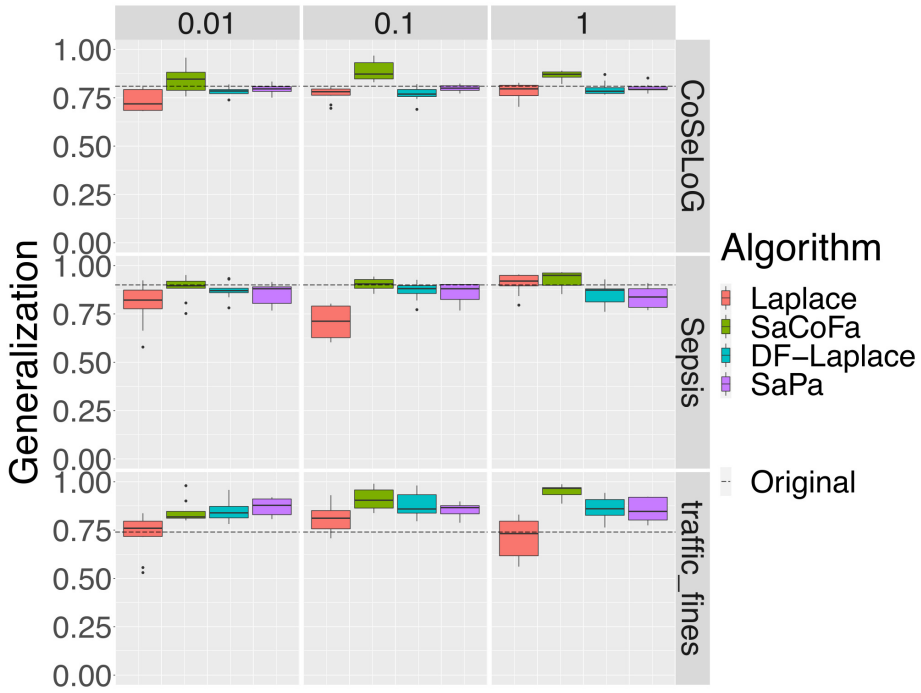


Figure 4: Generalization of discovered process models.

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

We present the results for the baseline that adopts the Laplace mechanism directly for the trace-variant distribution separately, in Fig. 7, to ensure the readability of the figures. This approach may produce results that have more than 200 times more traces than the original log, for small values of ϵ. Put

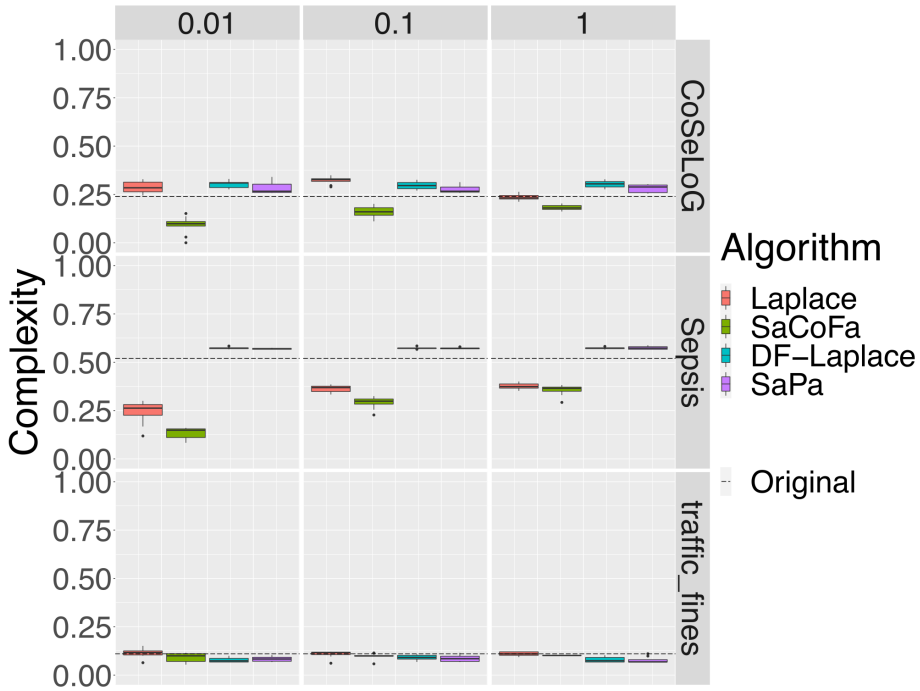


Figure 5: Complexity of anonymized event logs.

differently, the anonymized results do not provide any reliable information about the actual size of the original log. Considering this observation as well as the high variability in the results obtained with SaCoFa, we conclude that results generated by playing out an anonymized directly-follows distribution are beneficial in terms of preserving the number of traces of the log in the obtained trace-variant distribution.

Noise Insertion. Next, we turn to the presence of easily-recognizable noise. In Fig. 8, we show the percentage of behaviour that is classified as normal behaviour by the aforementioned anomaly detection technique. While all techniques achieve good results for the Traffic Fines dataset, there is a clear trend for the other two logs: the Laplace baseline technique produces much more noise that is directly recognizable as anomalous. Therefore, the traces introduced by SaCoFa or SaPa are more in line with the original process' behaviour.

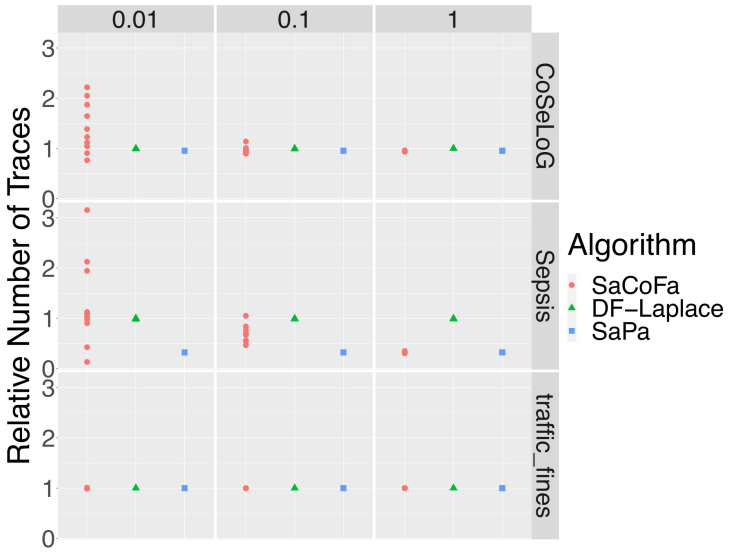


Figure 6: Relative size of anonymized logs compared to the original log

Study of the Exponential Mechanisms. Finally, we investigate technical aspects of SaCoFa and SaPa. We first turn to an experiment regarding the effects of semantics-aware pruning for SaCoFa. Fig. 9 shows the F-score of models discovered from the anonymized results obtained with and without pruning. Overall, semantics-aware pruning turns out to only be beneficial for the Traffic Fines log, which is the most structured one. For the less-structured logs, the F-score actually decreases in comparison to the approach without pruning. We attribute this observation to the significance of the rules used to separate harmful and harmless prefixes. Apparently, they are not always sophisticated enough to compensate for the additional variance introduced to the trace-variant distribution.

For SaPa, we investigated the effects of different values for the k-follows relation threshold, as employed in the score-function of the exponential mechanism. In Fig. 10, the F-score for different thresholds is shown. Overall, no general trend can be seen. We conclude that our technique is robust against changes in this parameter, while there is some evidence that a value of $k = 1$ is generally beneficial.

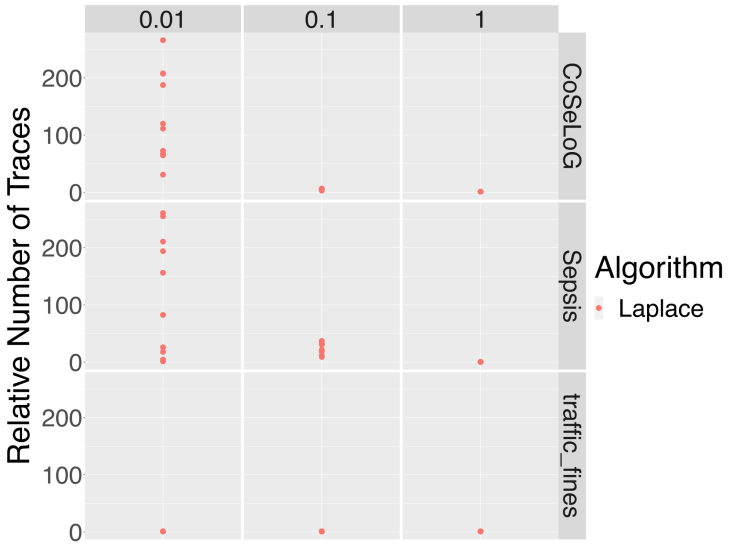


Figure 7: Relative size of anonymized logs compared to the original log

6. Discussion

We now turn to a discussion of different observations and characteristics of our proposed SaCoFa (Section 6.1) and SaPa (Section 6.2) approaches.

6.1. SaCoFa

Runtime aspects. As mentioned in Section 5.2, the employed parameters of SaCoFa must be selected carefully for trace-variant queries to complete in a reasonable time. Specifically, we observed that the anonymization procedure either terminated within seconds, or not all, for both SaCoFa and the Laplace trace-variant query. This reveals that there is a clear point when the prefix growth makes the trace-variant query intractable. Until now, this point is determined by step-wise altering the maximal variant length (k) and pruning parameters for a given ϵ. Nevertheless, we observe that SaCoFa can compute results for lower pruning thresholds faster than the Laplace baseline. However, to ensure a fair comparison, we used the same pruning parameters for all mechanisms in our experiments.

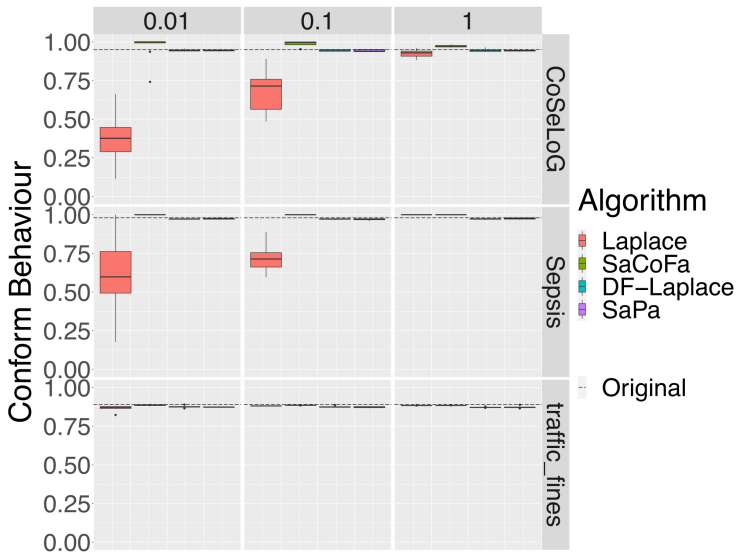


Figure 8: Relative frequency of normal behaviour in event logs.

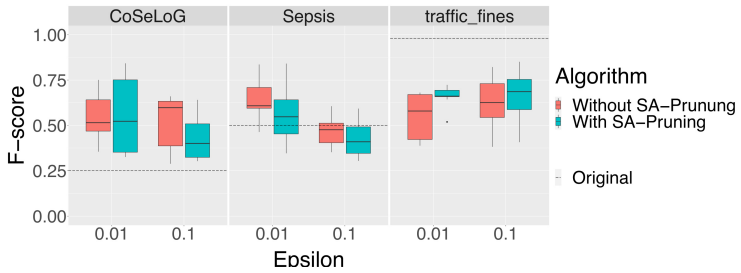


Figure 9: F-score for configurations with and without pruning of SaCoFa

Non-binary score functions. Beyond determining if a prefix is harmful or not, the behavioural appropriateness-based score can be used to quantify its degree of harmfulness. However, the sensitivity of the exponential mechanism depends on the maximal impact that a single case can have on the employed score function, i.e., on the function's maximal value (see Section 2). Therefore, if we define a score function that quantifies harmfulness in, e.g., the range $[0, 3]$, the query's sensitivity would thus be $\Delta f = 3$, instead of $\Delta f = 1$ for a binary assessment. Since the exponential mechanism inserts noise proportional to the sensitivity, such a non-binary classification leads to larger magnitudes of noise inserted. Therefore, the intended benefits obtained from quantifying harmfulness

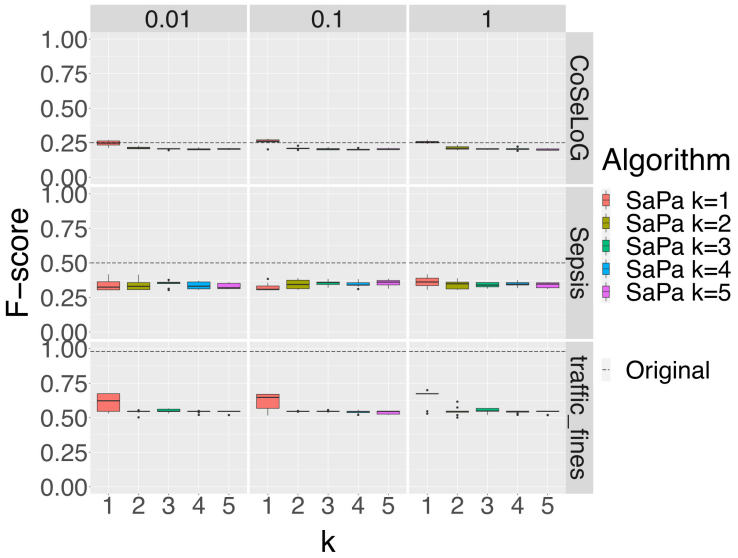


Figure 10: F-score for different configurations of SaPa

in a non-binary manner must outweigh the increased sensitivity for it to have a noticeable effect. While this did not appear to be the case in our current experiments, we believe that this approach may still be applicable for more sophisticated score functions, tailored to the specifics of the process at hand. In any case, it is important to consider this trade-off while selecting appropriate pruning parameter values.

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

Privacy Guarantee. The SaCoFa algorithm anonymizes the retrieved trace-variant distribution using noisy counts, drawn from a distribution adjusted to the differential privacy parameter ϵ and sensitivity of the function Δf (for trace variant queries, Δf=1), taken as input. Therefore, data anonymized in this manner is known to be protected by ϵ-differential privacy [13]. However, one can

argue that the actual protection may be stronger. The reason for this is that the pruning used by SaCoFa means that uncommon variants are less likely to be present in the final output of the algorithm, which reduces the difference in output between neighboring logs. Consequently, the pruning part of SaCoFa might even lead to a slightly higher privacy protection than chosen by the user.

6.2. SaPa

Unused directly-follows relations. One consequence of the play-out algorithm of SaPa is that some directly-follows relations are not considered for the final trace-variant distribution. As shown in Section 5, the approach was still able to generate trace-variant distributions with a reasonable utility. However, we believe it is important to notice this inherent information loss. It may be possible to minimize this loss through more sophisticated play-out algorithms, which involves some of the considerations discussed below.

Guided play-out algorithms. It is important to notice that the privacy guarantee for SaPa was given for the anonymized directly-follows distribution. Therefore, this distribution can be processed in any way possible, assuming that no other information about the event log is added. However, if the play-out algorithm would somehow be guided by information extracted from the original data, this would subsequently intervene with the given assumption, and thus with the privacy guarantee. Consequently, the privacy of the individuals would be harmed by more than allowed for by ϵ. Contrary, information extracted based on the labels of the activity, would not devalue the given privacy guarantee, since these informations are publicly revealed and therefore public knowledge. Therefore, for the development of play-out using additional information, it is necessary to extract the used information only from the anonymized distribution, i.e. by heuristics that minimize the number of unused directly-follows relations, or by usage of activity labels [37].

Sensitivity of the directly-follows query. The sensitivity, i.e., the maximum impact an individual may have on the query result needs to be known in advance

when applying differential privacy. For a trace-variant query, we assume this to be $\Delta f = 1$, i.e., one individual is only represented by exactly one trace. While this may not always be the case, it was shown that a strong privacy protection can still be given, without raising the sensitivity [21]. However, determining the maximum influence one individual may have on a directly-follows distribution, is more complex. Even if the assumption of one individual being represented by one trace holds, some directly-follows relations could be impacted by values far higher than 1 due to loops in the underlying process. Therefore, fixing the sensitivity to $\Delta f = 1$ might not be appropriate for processes involving a lot of loops. In such cases, it is either necessary to adjust the sensitivity accordingly or stick to SaCoFa, which by design is more robust.

Privacy Guarantee. In SaPa, the trace-variant distribution is retrieved from a differential private directly-follows distribution. Since differential private data is immune to post-processing, i.e., the guarantees remain when additional operations are applied afterwards, the final result of SaPa is therefore also protected by differential privacy [13]. Nonetheless, as discussed in the previous paragraph, setting the sensitivity Δf right for directly-follows queries is challenging and an incorrect setting can lead to a drop in the given privacy-guarantee.

7. Related Work

We introduced several approaches for control-flow anonymization that answers trace-variant queries, while guaranteeing differential privacy. The state of the art to derive a trace-variant distribution, and the related directly-follows graph, under differential privacy, uses the Laplace mechanism [14]. As discussed, this neglects a process' semantics, leading to potentially low data utility and noise that can be easily recognized. Another alternative approach aims at only publishing the trace-variants present in the original log, by oversampling [38]. Such an approach does not produce any new trace-variant, therefore risking the leakage of information, because the knowledge of a original sub-sequence of a trace-variant may leak the whole variant. Consequently, the approach is not comparable with

the presented approaches and was not considered in the evaluation. Besides these studies of trace-variant queries, Elkoumy et al. [39] further studied the relation between utility and risk, while the PRIPEL framework [40] uses trace-variant queries as a basis for privacy-preserving event log publishing.

Beyond differential privacy, the anonymization of event logs based on other privacy guarantees was studied. PRETSA [41] sanitizes event logs to ensure k-anonymity and t-closeness, which are guarantees based on the idea of grouping similar cases together. Furthermore, a process mining-specific extension of k-anonymity, called TLKC, was introduced in [42]. Previous work focused on improving the utility of these techniques through feature learning-based distance metrics [43]. Another group-based approach was introduced by Batista et al. [44], based on the uniformization of events within a group of individuals. The issue of continuously publishing anonymized event logs was studied in [45].

Approaching privacy preservation from the viewpoint of the analysis techniques used in process mining, it was shown how multi-party computation enables the construction of process models based on inter-organizational processes, without sharing the data between parties [46]. The same problem was also addressed to specialized hardware, that ensures a trusted execution[47]. Other approaches target privacy-aware role mining [48] and the establishment of privacy-aware process performance indicators through the enforcement of differential privacy [49].

The application of privacy-preserving process mining towards the healthcare domain was studied by Pika et al. [50] . Furthermore, several tools have been developed to support the application of privacy-preserving process mining such as ELPaaS [51], Shareprom [52] and PC4PM [53]. Further practical considerations of publishing anonymized event logs have been discussed in [54].

8. Conclusion

Targeting control-flow anonymization for business processes, we introduced two approaches to answer trace-variant queries in a privacy-preserving manner.

First, we introduced SaCoFa, an approach based on a prefix tree construction and the exponential mechanism. Unlike state-of-the-art techniques that leverage the Laplace mechanism and, hence, introduce random noise to achieve differential privacy, SaCoFa incorporates the semantics of the underlying process when inserting noise, achieving the same privacy guarantee. To this end, we introduced a score function that differentiates prefixes as being harmful or harmless, which then guides the anonymization.

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

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

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

angle to improve our work. Finally, while we have shown improvement in terms of the measured utility of anonymization through SaPa and SaCoFa it is still an open question how analysts can best utilize anonymized process mining results and deal with the injected noise.

Acknowledgements

This work was partially funded by the German Research Foundation (DFG), project 421921612, and the Leibniz Association as part of the Berlin Centre for Consumer Policies (BCCP). The authors are grateful to Felix Oesinghaus for his contributions to the implementation of the approaches and his support in preliminary experiments.

References

[1] W. Van Der Aalst, Process mining: Overview and opportunities, ACM Transactions on Management Information Systems (TMIS) 3 (2) (2012) 1–17.

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

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

[4] M. T. Wynn, A. Rozinat, W. M. P. van der Aalst, A. H. M. ter Hofstede, C. J. Fidge, Process mining and simulation, in: Modern Business Process Automation - YAWL and its Support Environment, Springer, 2010, pp. 437–457.

[5] I. Teinemaa, M. Dumas, M. L. Rosa, F. M. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Trans. Knowl. Discov. Data 13 (2) (2019) 17:1–17:57.

[6] S. A. Fahrenkrog-Petersen, N. Tax, I. Teinemaa, M. Dumas, M. de Leoni, F. M. Maggi, M. Weidlich, Fire now, fire later: alarm-based systems for prescriptive process monitoring, Knowl. Inf. Syst. 64 (2) (2022) 559–587.

[7] P. Voigt, A. Von dem Bussche, The EU general data protection regulation (GDPR), A Practical Guide, 1st Ed., Cham: Springer International Publishing.

[8] E. Goldman, An introduction to the California Consumer Privacy Act (CCPA), Santa Clara Univ. Legal Studies Research Paper.

[9] S. N. von Voigt, S. A. Fahrenkrog-Petersen, D. Janssen, A. Koschmider, F. Tschorsch, F. Mannhardt, O. Landsiedel, M. Weidlich, Quantifying the re-identification risk of event logs for process mining, in: CAiSE, Springer, 2020, pp. 252–267.

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

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

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

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

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

[15] S. A. Fahrenkog-Petersen, M. Kabierski, F. Rösel, H. van der Aa, M. Weidlich, SaCoFa: Semantics-aware control-flow anonymization for process mining, in: ICPM, IEEE, 2021, pp. 72–79.

[16] I. Wagner, D. Eckhoff, Technical privacy metrics: a systematic survey, ACM Computing Surveys (CSUR) 51 (3) (2018) 1–38.

[17] D. Team, et al., Learning with privacy at scale (2017). URL https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale

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

[19] Ú. Erlingsson, V. Pihur, A. Korolova, Rappor: Randomized aggregatable privacy-preserving ordinal response, in: ACM SIGSAC, 2014, pp. 1054–1067.

[20] F. McSherry, Privacy integrated queries: an extensible platform for privacy-preserving data analysis, in: U. Çetintemel, S. B. Zdonik, D. Kossmann, N. Tatbul (Eds.), ACM SIGMOD, ACM, 2009, pp. 19–30.

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

[22] F. McSherry, K. Talwar, Mechanism design via differential privacy, in: FOCS, IEEE, 2007, pp. 94–103.

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

[24] M. Weidlich, J. M. E. M. van der Werf, On profiles and footprints - relational semantics for petri nets, in: PETRI NETS, Springer, 2012, pp. 148–167.

[25] A. Polyvyanyy, M. Weidlich, R. Conforti, M. L. Rosa, A. H. M. ter Hofstede, The 4C spectrum of fundamental behavioral relations for concurrent systems, in: PETRI NETS, Springer, 2014, pp. 210–232.

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

[27] J. Buijs, Receipt phase of an environmental permit application process ('WABO'), CoSeLoG project (8 2014). doi:10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6.

[28] F. Mannhardt, Sepsis cases-event log, Eindhoven University of Technology. Dataset (2016) 227–228doi:10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

[29] M. M. De Leoni, F. F. Mannhardt, Road traffic fine management process (2015). doi:10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5.

[30] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: BPM Workshops, 2013, pp. 66–78.

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

[32] J. Munoz-Gama, J. Carmona, A fresh look at precision in process conformance, in: BPM, Springer, 2010, pp. 211–226.

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

[34] A. Augusto, J. Mendling, M. Vidgof, B. Wurm, The connection between process complexity of event sequences and models discovered by process mining, Information Sciences 598 (2022) 196–215.

[35] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation forest, in: ICDM, IEEE, 2008, pp. 413–422.

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

[37] H. van der Aa, A. Rebmann, H. Leopold, Natural language-based detection of semantic execution anomalies in event logs, Inf. Syst. 102 (2021) 101824.

[38] G. Elkoumy, A. Pankova, M. Dumas, Mine me but don't single me out: Differentially private event logs for process mining, in: ICPM, IEEE, 2021, pp. 80–87.

[39] G. Elkoumy, A. Pankova, M. Dumas, Privacy-preserving directly-follows graphs: Balancing risk and utility in process mining, arXiv preprint arXiv:2012.01119.

[40] S. A. Fahrenkrog-Petersen, H. van der Aa, M. Weidlich, PRIPEL: Privacy-preserving event log publishing including contextual information, in: BPM, Springer, 2020, pp. 111–128.

[41] S. A. Fahrenkrog-Petersen, H. van der Aa, M. Weidlich, PRETSA: event log sanitization for privacy-aware process discovery, in: ICPM, IEEE, 2019, pp. 1–8.

[42] M. Rafiei, W. M. van der Aalst, Group-based privacy preservation techniques for process mining, Data & Knowledge Engineering 134 (2021) 101908.

[43] F. Rösel, S. A. Fahrenkog-Petersen, H. v. d. Aa, M. Weidlich, A distance measure for privacy-preserving process mining based on feature learning, in: BPM Workshops, Springer, 2021, pp. 73–85.

[44] E. Batista, A. Solanas, A uniformization-based approach to preserve individuals' privacy during process mining analyses, Peer-to-Peer Networking and Applications (2021) 1–20.

[45] M. Rafiei, W. M. van der Aalst, Privacy-preserving continuous event data publishing, arXiv preprint arXiv:2105.11991.

[46] G. Elkoumy, S. A. Fahrenkrog-Petersen, M. Dumas, P. Laud, A. Pankova, M. Weidlich, Secure multi-party computation for inter-organizational process mining, in: BPMDS, Springer, 2020, pp. 166–181.

[47] M. Müller, A. Simonet-Boulogne, S. Sengupta, O. Beige, Process mining in trusted execution environments: Towards hardware guarantees for trust-aware inter-organizational process analysis, in: ICPM Workshops, 2021.

[48] M. Rafiei, W. M. van der Aalst, Mining roles from event logs while preserving privacy, in: BPM Workshops, Springer, 2019, pp. 676–689.

[49] M. Kabierski, S. A. Fahrenkrog-Petersen, M. Weidlich, Privacy-aware process performance indicators: Framework and release mechanisms, in: CAiSE, Springer, 2021, pp. 19–36.

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

[51] M. Bauer, S. A. Fahrenkrog-Petersen, A. Koschmider, F. Mannhardt, H. van der Aa, M. Weidlich, Elpaas: Event log privacy as a service, in: BPM Demos, Vol. 2420, CEUR-WS.org, 2019, pp. 159–163.

[52] G. Elkoumy, S. A. Fahrenkrog-Petersen, M. Dumas, P. Laud, A. Pankova, M. Weidlich, Shareprom: A tool for privacy-preserving inter-organizational process mining, in: BPM Demos, Vol. 2673, CEUR-WS.org, 2020, pp. 72–76.

[53] M. Rafiei, A. Schnitzler, W. M. P. van der Aalst, PC4PM: A tool for privacy/confidentiality preservation in process mining, in: BPM Demos, Vol. 2973, CEUR-WS.org, 2021, pp. 106–110.

[54] M. Rafiei, W. M. van der Aalst, Privacy-preserving data publishing in process mining, in: BPM Forum, Springer, 2020, pp. 122–138.