Towards Multiple Pattern Type Privacy Protection in Complex Event Processing Through Event Obfuscation Strategies

Saravana Murthy Palanisamy(✉)

University of Stuttgart, Stuttgart, Germany
saravanamurthypalanisamy@gmail.com

Abstract. For a Complex Event Processing (CEP) system to be widely accepted, mitigating leaks of private information is paramount. In CEP systems, often private information are revealed through patterns instead of single events. There are very few mechanisms that protect privacy at the level of patterns. However these mechanisms consider only sequential patterns, one of the common pattern types in CEP. But this is highly confined since there are other common pattern types like conjunction, negation etc. So as a first step towards multiple pattern type privacy protection, in this paper we present a hybrid pattern level privacy protection mechanism that considers three common pattern types: sequence, conjunction and negation. Our approach is based on three event obfuscation strategies: event reordering, event suppression and introduction of fake events to conceal private patterns on the one hand, while minimizing impact on useful non-sensitive information required by IoT services to provide a certain Quality of Service (QoS) on the other hand. Our evaluations over real-world datasets show that our algorithms are effective in maximizing QoS while preserving privacy.

Keywords: Privacy · Complex event processing · Event obfuscation

1 Introduction

The Internet of Things (IoT) envisions a world with billions of networked sensors connected to the internet. Studies show that over 2.5 quintillion bytes of data are produced every day from these sensors[14]. These data are often raw sensor values and need to be processed into meaningful information in order to be useful for IoT applications like smart homes, e-health etc. Complex Event Processing (CEP) is a famous state-of-the-art paradigm for processing streams of such raw basic events into meaningful information (called "complex events") using a set of processing rules [11]. For example, a fitness tracker can infer the activity "doing sports" via basic events: $speed > 10\,km/h$ and $heart\ rate > 100\,bpm$.

The challenges while inferring meaningful information is a double-edged sword. On one side meaningful information is required to offer a certain *Quality of Service* (QoS). There is loss in QoS if a CEP system does not detect

J. Garcia-Alfaro et al. (Eds.): DPM 2020/CBT 2020, LNCS 12484, pp. 178–194, 2020.
https://doi.org/10.1007/978-3-030-66172-4_11

some events that originally existed (false negatives) or detects some non-existing events (false positives). On the other side some complex events might be privacy-sensitive. Though a user needs complex events to be detected accurately, he/she might not want to share any privacy-sensitive events. Thus a privacy protection mechanism is necessary for users to conceal private events while preserving QoS.

Access control is one prominent technique for privacy protection in CEP. However, most access control mechanisms protect privacy only at the level of single attributes of data [2,16]. But sensitive information is often revealed through complex data *patterns* that usually span several attributes. For example, heart rate and blood pressure might not reveal any useful information when analysed separately, but might reveal a disease when combined. We call these privacy-sensitive patterns in the event stream that a user or data owner wants to protect *private patterns*, while those patterns that are necessary for CEP applications to offer services and are not privacy-sensitive *public patterns*. Hence a pattern-level access control mechanism is required with which private patterns can be *obfuscated* while preserving as many public patterns as possible (maximum QoS).

Pattern-based access control strategy for CEP systems were proposed in [13,18]. However both these mechanisms only deal with patterns defined as a sequence of events. Although a sequence operator is one of the most common operator types in CEP, there are other common types like conjunction, negation etc.[5]. Also there can be more than one type of private pattern appearing at the same time which a user wants to conceal. For instance, on a weekday a user has taken a day off from work and is shopping. In the area of location privacy, this scenario can be viewed as two private patterns: a) "shopping" deduced by a series of location events [17] which is a sequence pattern; b) "not at work" which is a negation pattern. A user might want both these patterns to be concealed.

So as a first step towards multiple operator privacy protection in this paper, we introduce *Multi-Operator Privacy Protection component* (MOP) based on Integer Linear Programming (ILP) that considers all three commonly used CEP operator types namely sequence, conjunction and negation. Our MOP is based on three event obfuscation strategies: event suppression, event reordering and introduction of fake events, to conceal private patterns while maximizing QoS.

Overall, we make the following contributions in this paper: (1) We propose a baseline event obfuscation approach MOP, to maximize the utility of obfuscated event stream. (2) We introduce two extensions of the baseline approach working against two different adversary models. (3) For the evaluation of these approaches, we define the two adversary models that consider background knowledge of event distributions etc. gained from event histories.

The rest of this paper is organized as follows. In Sec. 2 we discuss related works followed by system model and problem statement in Sec. 3. In Sec. 4, we present our event obfuscation approaches that strive to maximize the utility of obfuscated event streams. In Sec. 5, we describe how an adversary uses observations learned from history data to reveal event obfuscations and describe our evaluation results, before concluding the paper in Sec. 6.

2 Related Works

Various access control mechanisms have been proposed to ensure privacy in event processing systems. However most of these works protect privacy at the level of attributes, ensuring that certain attributes in the event stream are only accessible to authorized parties [2,10]. However this is highly confined, since some attributes are either accessible or not at all, irrespective of whether it is part of a private or public pattern.

Another branch of privacy protection in event processing systems is differential privacy [4,8] and zero-knowledge privacy guarantees [15]. Though these mechanisms promise provable privacy guarantees, these works either protect privacy at the level of individual events/attributes or they protect privacy for individual users whose data are part of a dataset from a large population of users. The goal of our work is different in the sense that we try to achieve pattern-level privacy, considering data from a single user rather than a population of users.

Very few works [7,18] have been published that protect privacy at the level of patterns for CEP systems. Wang et al. [18] proposed a pattern-based access control strategy based on event suppression for sequence patterns. This approach conceals patterns by suppressing events that are part of private patterns while maximizing utility. In [13], Palanisamy et al. proposed an approach based on event reordering rather than suppression, but again only for sequence patterns. To the best of our knowledge there are not any research works that ensure pattern level privacy for CEP systems that consider multiple pattern types.

3 System Model and Problem Statement

In this section, we introduce our system model, which also includes our assumptions about the adversary who tries to detect private patterns. Moreover, we use the utility metric defined in [13] that defines the utility of an obfuscated event stream, which is then used as an objective in our problem statement.

3.1 System Model

Our system consists of the following components (cf. Figure 1): producers, consumers, CEP middleware in addition to Multi-Operator Privacy protection.

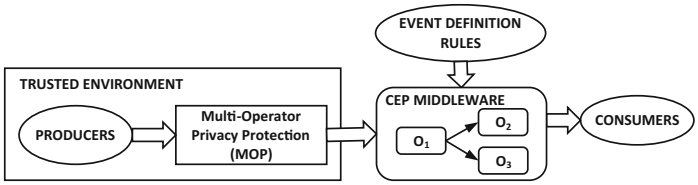


Fig. 1. CEP System + Multi-operator privacy protection component

Like in a typical CEP system [11], *Producers* generate basic events as input to the system. A CEP system could have multiple producers, but for this paper we assume that event streams from different producers are merged together into a single event stream in the order of timestamps like in [18]. Of course it is not possible for the users to know the exact event patterns that reveal sensitive information. Thus a separate system to translate the privacy requirements of a user into event patterns is necessary and this is a separate area of research [12]. For our work we assume that the user uses such a system to provide the necessary event patterns to be concealed. The *CEP middleware* extracts high level information (complex events) from basic events which are then forwarded to the *consumers* (e.g. an IoT service provider). We use a standard CEP middleware [6] which is a set of interconnected operators (O_x in Fig. 1). Operators are computing nodes that transforms input event stream into one or more outgoing event streams based on a set of predefined rules. The cooperative processing of several such operators result in a complex event. Generally, operators do not consider the complete (theoretically infinite length) event stream, instead process the event stream in smaller frames within a limited time or length called *windows*.

In this paper we consider three common operator types: *sequence*, *negation*, and *conjunction* [5]. A sequence operator captures a pattern in which a set of events arrive in a specific order in a window. An example sequence pattern denoting unhealthy behaviour of a diabetes patient (i.e., a private pattern) that should be hidden to an untrusted party (say health insurance company of the user) could be: $SEQ(Eating\_Sugar, High\_Blood\_Sugar\_Level, Insulin\_Intake)$. A conjunction operator queries the occurrence of a set of events within a window in any order and an example private pattern for the same user could be: $AND(Eating\_Sugar, No\_Insulin\_Intake)$. A negation operator queries the absence of a particular event within a specified window and an example private pattern could be: $NOT(Insulin\_Intake)$.

Like in [13,18], we also assume that the CEP middleware and the consumers are untrusted and are operated by another party other than the user (e.g., an IoT service provider). So private patterns from producers should be concealed before they are forwarded to the CEP or consumer. Thus we place our *Multi-Operator Privacy Protection component* (MOP) between producers and CEP and forward all producer events via MOP as shown in Fig. 1. The MOP performs three obfuscation strategies namely event reordering, event suppression and introduction of fake events to hide sequence, conjunction and negation type private patterns respectively. Both, producers and MOP, run in a trusted execution environment (e.g. smart phone of a user, private fog node etc.).

It is also important to specify the assumptions about the adversary that tries to compromise the privacy of users by detecting private patterns. We assume a "honest-but-curious" adversary which is a common model in privacy. In other words, one of the non-trusted components (CEP middleware or consumer) might be the adversary. In this case, the adversary is not able to observe the original event stream, but can observe the whole obfuscated event stream as sent by the

MOP. We also assume that the adversary has some background knowledge about the original event stream learned from external sources (say publicly available event streams).

3.2 Problem Statement

Informally, the problem solved by our approach is to preserve privacy by performing the three obfuscations while maximizing QoS. This is not a trivial problem, since there might be different combinations of obfuscations each having varying impact on QoS. We measure QoS in terms of false positive patterns (public patterns introduced by obfuscations that never happened) and false negative patterns (actual public patterns destroyed by obfuscations). In order to define the impact of obfuscations more precisely, we use the *utility metric* proposed in [13] as given in Equ. 1. However for our work the three terms of this utility metric now include all three pattern types as opposed to only sequence type patterns in case of [13]. The utility metric (U) is given as follows:

$$U = \Sigma_{i=1}^{\#\ of\ matched\ true\ public\ Patterns} w_i \quad - \quad \Sigma_{j=1}^{\#\ of\ matched\ false\ positives} w_j \\ -\Sigma_{k=1}^{\#\ of\ matched\ private\ patterns} w_k \tag{1}$$

The second term in the equation is to introduce a negative penalty for each false positive. Negative penalties for false negatives are deducted already, since they are not included in matched public patterns thus reducing the actual utility metric. Thus, the first two terms increase the utility for each true positive match of a public pattern and decrease the utility for each false positive or false negative match. The weights w_i and w_j shown in Equ. 1 show the relative importance of different public patterns onto QoS. Matched private patterns are considered in the third term. Including private patterns in the utility metric rather than making it a hard constraint (no private pattern matches) enables us to obtain a trade-off between privacy and QoS, i.e., revealing some private information for more public pattern matches. Here, weight w_k of a private pattern k is given as:

$$w_k = (\Sigma w_i + 1) * cp_k \tag{2}$$

Tuning parameter cp_k enables to leverage privacy for QoS by specifying a criticality percentage for private pattern k. If we set the private pattern criticality to 100%, i.e., $cp_k = 1$, then a single match of that private pattern k would outweigh the effect of all public pattern matches such that, no private patterns of that type will be revealed (100% privacy). For $cp_k = 0$ that type of private pattern is always revealed. The problem now is to find a strategy that performs the three obfuscations and maximizes utility while making it (ideally) impossible for the adversary to detect concealed private patterns. Solving this problem requires some assumptions about the background knowledge of the adversary to understand what possibilities the adversary has to detect event obfuscations. We consider two kinds of adversary model: deterministic and probabilistic.

Deterministic Adversary model: This type of adversary requires 100% "confidence" that a private pattern is concealed. One example for such an adversary could be a car manufacturer who is curious to detect a concealed private pattern (say bad driving behaviour of the user) that caused damage to the vehicle, with 100% confidence to deny a warranty claim. The adversary in this case employs three attacks one for each obfuscation mechanism.

Causal order constraint attacks rely on knowledge about causal relationship between events. We say the order of two events e_1 and e_2 is causally constrained if e_2 must never happen before e_1. Reordering e_1 and e_2 could immediately be detected as a violation of this constraint known to the adversary.

Periodic event constraint attacks rely on knowledge about periodic events. For instance, in a e-health scenario the event "blood sugar level" is recorded every 30 mins. Suppressing such events could be detected immediately as a violation of this constraint.

Infeasible event constraint attacks are based on knowledge about events that are impossible to happen in a window. For example, the location event "at mall" is infeasible after closing hours. Introducing such fake events could be detected immediately as a constraint violation.

Probabilistic Adversary model: Here, it is sufficient for the adversary to detect a concealed private pattern with a confidence (γ) $< 100\%$. An example adversary could be a car insurance company who might increase the insurance premium, even if a concealed private pattern about the user's bad driving behaviour is revealed with 80% confidence. The adversary in this model employs statistical attacks, in addition to those from the deterministic adversary.

Statistical attacks consider the inter-arrival time distribution of events, either between same or different types of events. For instance, by analysing a publicly available data set, an adversary might know that, in 90% of all cases the time between two insulin injections for patients is around 2 h. Suppressing an "insulin injection" event would lead to a difference of 4 h between two injections indicating the adversary that an event has been suppressed with 90% confidence. Thus, besides maximizing utility, obfuscations are to be performed such that it is unlikely to detect event obfuscations performed with 100% confidence for deterministic adversaries and below a certain confidence threshold (γ) for probabilistic adversaries.

4 Event Obfuscation Approaches

In this section, we present our two event obfuscation approaches: Counter Deterministic Attack (CDA) and Counter Probabilistic Attack (CPA) obfuscation strategies that conceals private patterns against deterministic and probabilistic adversaries respectively. Both the approaches are extensions of a *baseline approach* based on Integer Linear Programming (ILP) which we will present first in the next sub-section followed by the two extensions.

4.1 Baseline ILP Approach

The primary goal of this ILP approach is to find an optimal combination of the three event obfuscation strategies that maximizes utility as defined by Equ. 1. The ILP is invoked on a full window of events, whenever this window contains at least one private pattern of any type. Of course, those windows without private patterns can be forwarded as it is. The problem of finding an optimal combination of the event obfuscation strategies is translated as an ILP with the utility metric as objective.

ILP Formulation:- Parameters: We now derive the parameters to formulate our ILP.

Table 1. ILP parameters

Parameters	Parameter description
$\mathcal{N}$	Number of events in the window
$\mathcal{T}_N$	Set of timestamps of event instances in the window
$\mathcal{N}^S_{TQ}$, $\mathcal{N}^C_{TQ}$, $\mathcal{N}^N_{TQ}$, $\mathcal{N}^S_{FQ}$, $\mathcal{N}^C_{FQ}$, $\mathcal{N}^N_{FQ}$	Set of true and false positives for sequence, conjunction and negation type *public* patterns
$\mathcal{N}^S_{TP}$, $\mathcal{N}^C_{TP}$, $\mathcal{N}^N_{TP}$, $\mathcal{N}^S_{FP}$, $\mathcal{N}^C_{FP}$, $\mathcal{N}^N_{FP}$	Set of true and false positives for sequence, conjunction and negation type *private* patterns
$\mathcal{W}^S_{TQ}$, $\mathcal{N}^C_{TQ}$, $\mathcal{N}^N_{TQ}$, $\mathcal{W}^S_{FQ}$, $\mathcal{N}^C_{FQ}$, $\mathcal{N}^N_{FQ}$	Set of weights for true and false positives for sequence, conjunction and negation type *public* patterns
$\mathcal{W}^S_{TP}$, $\mathcal{N}^C_{TP}$, $\mathcal{N}^N_{TP}$, $\mathcal{W}^S_{FP}$, $\mathcal{N}^C_{FP}$, $\mathcal{N}^N_{FP}$	Set of weights for true and false positives for sequence, conjunction and negation type *private* patterns

We define $\mathcal{N}$ as number of events in the window on which our ILP is invoked. We introduce another parameter set $\mathcal{T}_N$ of size $\mathcal{N}$ representing arrival timestamps of event instances in that window. $\mathcal{N}^S_{TQ}$, $\mathcal{N}^C_{TQ}$, and $\mathcal{N}^N_{TQ}$ represent list of matched sequence, conjunction, and negation type *true positive* public patterns in the window before obfuscation. $\mathcal{N}^S_{FQ}$, $\mathcal{N}^C_{FQ}$, and $\mathcal{N}^N_{FQ}$ represent list of sequence, conjunction and negation type *false positive* public patterns. Here TQ and FQ represent true and false positive public patterns. Similar to public patterns, we have $\mathcal{N}^S_{TP}$, $\mathcal{N}^C_{TP}$, $\mathcal{N}^N_{TP}$, $\mathcal{N}^S_{FP}$, $\mathcal{N}^C_{FP}$, and $\mathcal{N}^N_{FP}$ representing list of true and false positive private patterns for the three operator types. Here, TP and FP represent true and false positive private patterns. The above 12 parameters are collectively called *pattern parameters*. As described in Equ. 1, we require positive weights signifying relative importance between public patterns and negative weights for false positives and private patterns. Thus we define 12

Table 2. ILP variables

Variables	Variable description
$\mathbf{V}^S_{TQ}$, $\mathbf{V}^C_{TQ}$, $\mathbf{V}^N_{TQ}$, $\mathbf{V}^S_{FQ}$, $\mathbf{V}^C_{FQ}$, $\mathbf{V}^N_{FQ}$	Set of decision variables for true and false positives for sequence, conjunction and negation type *public* patterns
$\mathbf{V}^S_{TP}$, $\mathbf{V}^C_{TP}$, $\mathbf{V}^N_{TP}$, $\mathbf{V}^S_{FP}$, $\mathbf{V}^C_{FP}$, $\mathbf{V}^N_{FP}$	Set of decision variables for true and false positives for sequence, conjunction and negation type *private* patterns
$\mathbf{O}_{NN}$	Matrix for all $\mathcal{N} \times \mathcal{N}$ ordered pairs in the window
$\mathbf{X}_N$	Set of variables representing change in timestamps
$\mathbf{E}_N$	Set of decision variables representing event suppressions
$\mathbf{I}_I$	Set of decision variables representing event introductions

sets of parameters representing weights ($\mathcal{W}^Y_{XX}$) for all three pattern types that correspond to each of the pattern parameters as shown in the last two rows of Table 1. These weights are the objective coefficients of our ILP.

Variables: We now define variables required for our ILP formulation (cf. Table 2). We define 3 sets of binary decision variables: $\mathbf{V}^S_{TQ}$, $\mathbf{V}^C_{TQ}$, and $\mathbf{V}^N_{TQ}$, one for each pattern type and each variable in a set represent a match for a true positive public pattern of that type. Similarly $\mathbf{V}^S_{TP}$, $\mathbf{V}^C_{TP}$, and $\mathbf{V}^N_{TP}$ represent match for true positive private patterns. Similarly for false positives, we introduce another 6 sets of decision variables representing matched false positive public and private patterns for each operator type: $\mathbf{V}^S_{FQ}$, $\mathbf{V}^C_{FQ}$, $\mathbf{V}^N_{FQ}$ and $\mathbf{V}^S_{FP}$, $\mathbf{V}^C_{FP}$, $\mathbf{V}^N_{FP}$. Among the false positive variables, those that represent sequence and conjunction patterns are positive integer variables. This is because there might be multiple false positive matches for the same pattern after obfuscation (since there can be multiple instances of the same event type) which is not known a priori in case of sequence and conjunction patterns which in turn should be considered for determining the optimal solution. However it is sufficient to define binary variables for false positive negation type patterns since there cannot be multiple false positive matches for the same negation type pattern in a window. The sizes of these 12 variable sets are determined by the cardinality of their corresponding *pattern parameters* as given in Table 1.

All the binary decision variables mentioned above have a common interpretation and a sample interpretation is given as: $\mathbf{V}^S_{TQ}[i] = 1$ if i^{th} pattern is matched in that window and 0 otherwise (Here $i \in \{1, 2, ...|\mathcal{N}^S_{TQ}|\}$). Similarly all integer decision variables have the same interpretation and a sample interpretation is given as $\mathbf{V}^S_{FQ}[j]$ is ≥ 1, one for every match of that pattern and 0 if that pattern has no match at all in that window (Here $j \in \{1, 2, ...|\mathcal{N}^S_{FQ}|\}$).

The linear combination of the weights and above mentioned decision variables form the utility function and is also the objective function of our ILP. For clarity, we group these 12 decision variable sets into 4 sets $\mathbf{V}_{TQ}$, $\mathbf{V}_{FQ}$, $\mathbf{V}_{TP}$, $\mathbf{V}_{FP}$ representing true positive public & private and false positive public & private patterns by grouping variables of all three pattern types together into one. The weight parameters are also similarly grouped: $\mathcal{W}_{TQ}$, $\mathcal{W}_{FQ}$, $\mathcal{W}_{TP}$, $\mathcal{W}_{FP}$. Now the objective function is written as:

$$maximize \sum_{i=1}^{|\mathcal{N}_{TQ}|} \mathcal{W}_{TQ} * \mathbf{V}_{TQ} + \sum_{j=1}^{|\mathcal{N}_{FQ}|} \mathcal{W}_{FQ} * \mathbf{V}_{FQ} + \sum_{k=1}^{|\mathcal{N}_{TP}|} \mathcal{W}_{TP} * \mathbf{V}_{TP} + \sum_{m=1}^{|\mathcal{N}_{FP}|} \mathcal{W}_{FP} * \mathbf{V}_{FP} \tag{3}$$

We will now describe the auxiliary variables necessary for our ILP. For a sequence pattern (say SEQ(A,B,C)) to be matched in a window, all ordered event pairs [SEQ(A,B), SEQ(B,C)] of that pattern should be matched. An ordered event pair is said to be matched if the two events occur in the same sequence after obfuscation. Thus we introduce a binary variable matrix $\mathbf{O}_{NN} \in \{0, 1\}^{\mathcal{N} \times \mathcal{N}}$ that represent whether the ordered pairs are matched in the window, and a set of bounded integer variables $\mathbf{X}_N$ of size $\mathcal{N}$ that represents the change in arrival timestamps after obfuscation for all event instances in that window. The relation between these two variables is written as a constraint given by:

$$\mathbf{O}_{NN}[i][j] = \begin{cases} 1 \rightarrow \text{if } \mathcal{T}_N[j] + \mathbf{X}_N[j] > \mathcal{T}_N[i] + \mathbf{X}_N[i] \\ 0 \rightarrow \text{otherwise} \end{cases} \tag{4}$$

where $i, j \in \{1, 2, ...\mathcal{N}\}$. This means that if an event instance e_i happens before e_j, then $\mathbf{O}_{NN}[i][j] = 1$.

For a conjunction pattern to be matched in a window, all instances of event types that constitute a conjunction pattern should occur in that window. Thus, we define a set of binary variables $\mathbf{E}_N$ of size $\mathcal{N}$ such that $\mathbf{E}[i] = 0$ if i^{th} event instance in that window is suppressed and is 1 otherwise (Here $i \in \{1, 2, ...\mathcal{N}\}$).

For a negation pattern to match, no event instance of that event type in the negation pattern should occur in the window. To conceal a negation private pattern, it is necessary to introduce a fake event instance of the event type that constitutes the pattern. Thus we define another binary decision variable set $\mathbf{I}_I$ of size $|\mathcal{N}_{TP}^N|$ for each negation type private pattern in addition to $\mathbf{E}_N$ defined above. $\mathbf{I}_I$ is interpreted as $\mathbf{I}_I[i] = 1$ if that i^{th} event is introduced fake and 0 otherwise here $i \in \{1, 2, ...|\mathcal{N}_{TP}^N|\}$. This means that additional events might be introduced in the window and so we extend $\mathbf{O}_{NN}$ to $\mathbf{O}_{NN}^*$, whose size will then be $(\mathcal{N} + |\mathcal{N}_{TP}^N|) \times (\mathcal{N} + |\mathcal{N}_{TP}^N|)$. Also $\mathcal{T}_N$ and $\mathbf{X}_N$ are changed to $\mathcal{T}_N^*$ and $\mathbf{X}_N^*$ whose sizes are now changed to $\mathcal{N} + |\mathcal{N}_{TP}^N|$.

Constraints: Now, we will explain the constraints, that translate the requirement for the event obfuscation problem. We classify these constraints into three categories: sequence, conjunction, and negation constraints.

Sequence constraints ensure that a sequence pattern is matched only if all ordered pairs of that pattern are matched. Here, all true positive sequence patterns (both public & private) have similar constraints and a sample constraint is given by

$$\forall \mathcal{N}_{TQ}^S : \quad IF(AND(\mathbb{M}_{TQ}^S[i])) \ THEN \ (\mathbf{V}_{TQ}^S[i] = True) \tag{5}$$

where $\mathbb{M}_{TQ}^S[i] \subset \mathbf{O}_{NN}$ consists set of all ordered pairs that correspond to i^{th} sequence pattern in $\mathcal{N}_{TQ}^S$. For false positive sequence patterns (both public and private), since we do not know a priori the number of matches, it is necessary to consider all possible combinations of ordered pairs that might lead to a match. A sample constraint for a false positive sequence pattern is given as:

$$\forall \mathcal{N}_{FQ}^S : \quad \mathbf{V}_{FQ}^S[i] = \sum_{j=1}^{|\mathbb{M}_{FQ}^S[i]|} AND(\mathbb{M}_{FQ}^S[i][j]) \tag{6}$$

where $\mathbb{M}_{FQ}^S[i]$ contains list of all ordered pair combination sets of the i^{th} sequence pattern in $\mathcal{N}_{FQ}^S$ and $\mathbb{M}_{FQ}^S[i][j] \subset \mathbf{O}_{NN}^*$. The above two constraints ensure correctness of sequence pattern matches in terms of ordered pairs. Since we might also perform suppression, when a suppressed event is part of a sequence pattern, that sequence pattern is also suppressed. Thus to ensure correctness of a sequence pattern match in terms of suppressed events, we add another set of constraints for all sequence patterns, and a sample constraint is given as:

$$\forall \mathcal{N}_{TQ}^S : \quad IF(AND(\mathbb{U}_{TQ}^S[i]) \ THEN \ (\mathbf{V}_{TQ}^S[i] = True) \tag{7}$$

where $\mathbb{U}_{TQ}^S[i] \subset (\mathbf{E}_N \cup \mathbf{I}_I)$ consists set of all events that are part of sequence pattern i. $\mathbb{U}$ contains set of all events for every true positive sequence pattern and list of all sets of event combinations for every false positive sequence pattern.

Conjunction constraints ensure that a conjunction pattern is matched only if all event instances of that conjunction pattern are not suppressed. For true positive conjunction patterns a sample constraint is given by:

$$\forall \mathcal{N}_{TQ}^{C}: \quad IF(AND(\mathbb{U}_{TQ}^{C}[i]))\ THEN\ (\mathbf{V}_{TQ}^{C}[i] = True) \tag{8}$$

where $\mathbb{U}_{TQ}^{C}[i] \subset \mathbf{E_N}$ contains set of all events that are part of the conjunction pattern i. This constraint is common for all true positive conjunction patterns (both private and public). For false positive conjunction patterns (both public and private) a sample constraint is given by:

$$\forall \mathcal{N}_{FQ}^{C}: \quad \mathbf{V}_{FQ}^{C}[i] = \sum_{j=1}^{|\mathbb{U}_{FQ}^{C}[i]|} AND(\mathbb{U}_{FQ}^{C}[i][j]) \tag{9}$$

where $\mathbb{U}_{FQ}^{C}[i]$ contains list of all event combination sets that correspond to the i^{th} conjunction pattern in $\mathcal{N}_{FQ}^{C}$ and $\mathbb{U}_{FQ}^{C}[i][j] \subset (\mathbf{E}_N \cup \mathbf{I}_I)$.

Negation constraints ensure that a negation pattern is matched only if an event instance of a particular event type does not occur in that window. It is not necessary for any additional constraints for true positive negation type public patterns since those events will not be introduced. Only those events that are part of true positive negation type private patterns might be introduced. Thus the constraint for negation type private patterns is given by:

$$\forall \mathcal{N}_{TP}^{N}: \quad IF(\mathbf{I}_I[i] == True)\ THEN\ (\mathbf{V}_{TP}^{N}[i] = False) \tag{10}$$

Event suppression might introduce a false positive negation pattern. Formally for a false positive negation pattern to be true, it is necessary that all event instances of that event type that define the negation pattern are suppressed. A sample constraint for a negation type false positive pattern is given as:

$$\forall \mathcal{N}_{FQ}^{N}: \quad IF(NOT(OR(\mathbb{U}_{FQ}^{N}[i])))\ THEN\ (\mathbf{V}_{FQ}^{N}[i] = True) \tag{11}$$

where $\mathbb{U}_{FQ}^{N}[i] \subset \mathbf{E}_N$ contains set of all event instances of that event type that constitute i^{th} negation pattern. The above constraint is common for all false positive negation patterns (both public and private).

The baseline approach described above aims at maximizing utility but does not consider all adversarial impacts. In the next subsections we will describe how this baseline approach is extended to the CDA and CPA obfuscation strategies that protect against deterministic and probabilistic adversaries respectively.

4.2 Counter Deterministic Attack Obfuscation (CDA)

To ensure protection against the deterministic adversary, it is sufficient if the approach does not perform any impossible obfuscations. So in this approach we introduce constraints such that our ILP neither reorders causally ordered pairs nor suppresses periodic events nor introduces fake infeasible events.

4.3 Counter Probabilistic Attack Obfuscation (CPA)

Probabilistic adversaries consider statistical attacks in addition to attacks of deterministic adversaries. Thus it is necessary to reduce the confidence level of the adversary below a certain confidence (γ) as described in Sect. 3. We propose such an approach that reduces adversary confidence by introducing fake event-obfuscations analogous to real ones such that obfuscations performed for concealing private patterns become indistinguishable from the *pseudo-obfuscations*. To achieve this, it is necessary to consider three factors:

1. *Introduce pseudo-obfuscations*: To introduce pseudo-obfuscations, we define pseudo-private patterns similar (but not exactly the same) to real private patterns. This is done by defining pseudo-private patterns with a combination of events that are predominantly part of real private patterns. We add these pseudo-patterns to the list of private patterns to conceal in addition to actual private patterns. The ILP, thus treats both pseudo and real private patterns in the same manner and hence the output obfuscations introduced are indistinguishable and thus the confidence of the adversary is reduced.
2. *Number of pseudo-obfuscations to be introduced*: To reduce the confidence of the adversary below γ, by proportionality principle it is necessary to introduce atleast $\kappa = ((\rho/\gamma) - 1)\%$ obfuscations where ρ is the *prior probability* the adversary has, before we introduce pseudo-obfuscations. If ρ is below the confidence threshold, it is not necessary to introduce any pseudo-obfuscations.
3. *Selection of windows for pseudo-obfuscations*: For selecting windows to introduce pseudo-obfuscations one simple way is to uniformly distribute the pseudo-private patterns over all the windows. But utility of those windows that already contain private patterns might get affected. This is because higher the number of private patterns to be concealed, higher the probability that concealing a private pattern would also affect a public pattern. So those windows that do not have any private pattern matches are selected for introducing pseudo-private patterns.

Overall, the idea is to introduce pseudo-private patterns along with real private patterns only to selected windows such that the obfuscations are revealed with less than $\gamma\%$ confidence.

5 Evaluation Results

In this section, we evaluate our two obfuscation approaches with respect to their ability to achieve indistinguishability against adversaries with minimum impact

onto public patterns (preserving QoS). As all state-of-the-art approaches are based on single operator types, it is not fair to compare our approach in terms of either privacy or utility for our evaluations. This is because utility and privacy are no longer the same between single and multi operator approaches even for the same window of events. Moreover, obfuscating one operator type pattern might introduce false positives or negatives of a different pattern type. Such interdependencies do not exist in case of single operator approaches.

5.1 Evaluation Setup

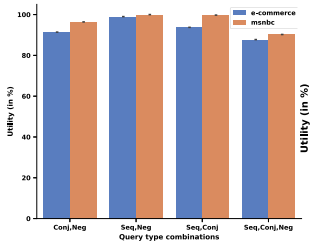
For the performance evaluation, we used a commodity server with AMD Ryzen 5-2500U processor (6 cores at 2.0 GHz) and 16 GB RAM. We implemented our obfuscation algorithms in Python using Gurobi as the ILP solver.

Datasets: To evaluate our approach we used two publicly available real-world datasets. The first dataset is an online retail dataset [3] that contains all transactions between 01/12/2010 and 09/12/2011 of a UK based online retailer selling all-occasion gifts. It includes 20,000 transactions and 500,000 purchased items from among 3,200 different products with timestamps and customer ids. We selected 50 most popular sequence, conjunction (of length 2 to 4) and negation patterns altogether as public and private patterns of varying length (e.g. A user hosting a Christmas Party deduced by the pattern: AND(Party Cone Christmas Decoration, Traditional Christmas Ribbons, Party invites Christmas)). The window size for these patterns vary from 5–800 events.

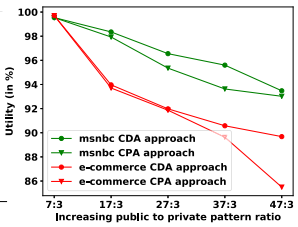
The second dataset is a msbnc web page visit dataset [1]. The dataset includes anonymous web page visits of users who visited msnbc.com in September 1999. Each window in the dataset corresponds to page views of a user during that day. Each event in the window corresponds to a user's request for a web page. We use the page visits of 20000 users. Similar to the e-commerce dataset, we searched and selected 25 most popular sequence, conjunction and negation patterns of varying length. Window size is the number of page visits of a user in a day and it varies between 3 and 400 pages. Such a dataset might reveal private information about the user like travel destinations, favourite TV-shows etc.

5.2 Adversary Model

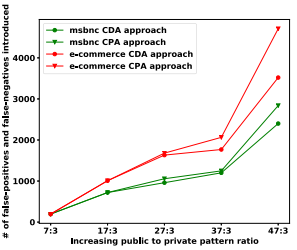
To evaluate the effectiveness of our approaches in protecting privacy, we first need to elaborate on the two adversary models. In our system model (Sec. 3), we have assumed that the deterministic adversary has information about causal, periodic and infeasible event constraints. We also assumed that the probabilistic adversary has background knowledge about statistical information about true (typical) inter-arrival time distribution of events on the modified event stream.



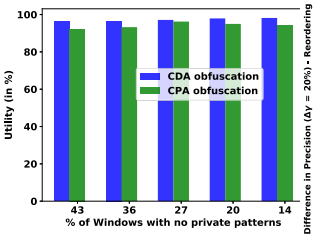
(a) Utility for different combination of pat types



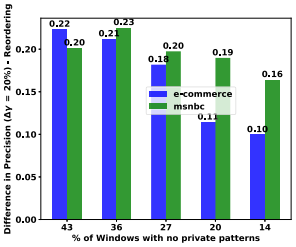
(b) Utility vs # of public patterns



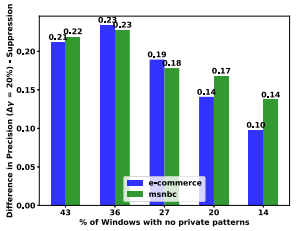
(c) False pos and False neg vs # of pub. patterns



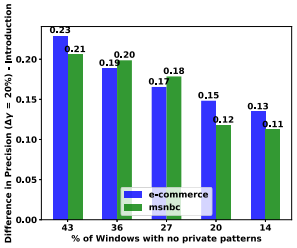
(d) Comparison of Utility between CDA and CPA



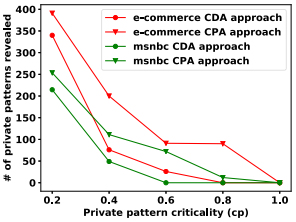
(e) Adversary Precision drop for Reordering-CPA



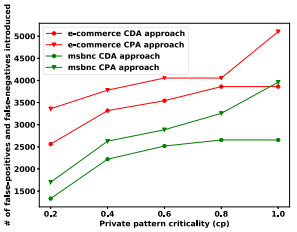
(f) Adversary Precision drop for Suppression-CPA



(g) Adversary Precision drop for fake event introduction-CPA



(h) # of private patterns vs criticality



(i) # of false-positives and false-negatives vs criticality

Fig. 2. Evaluations results

Statistical attacks: We assume that from publicly available data the adversary knows the true mean value $\mu(A)$ and true standard deviation $\sigma(A)$ of the inter-arrival time distribution for events of event types say A, B, C, etc. So now from observing the obfuscated event stream, for an event instance say A_i, the adversary can calculate the so-called *z-score* [9] $z(A_{i-1}, A_i) = \frac{t(A_i) - t(A_{i-1}) - \mu((A))}{\sigma((A))}$, i.e., the number of standard deviations by which the observed inter-arrival time is below or above the true mean inter-arrival time. When the adversary observes a potentially suppressed event say A_i in the event stream, he calculates the *Suppression Indicator (SI)* $= 1 - p_z(A_{i-1}, A_{i+1})$ where p_z is the probability value corresponding to the z-score of that event instance $(z(A_{i-1}, A_{i+1}))$.

p_z values corresponding to z-scores vary based on arrival-time distribution of events. Besides it is sufficient to calculate *SI* for each event of a conjunction pattern separately and then combine them, since event suppression is also done at the level of single events. If *SI* is above the confidence threshold (γ), then the adversary assumes that a private pattern is concealed in that window. Similar calculation is employed when the adversary observes a potential fake introduced event where the z-scores would be negative instead.

Statistical attacks for Sequence patterns: To this end, we assume that for each ordered pair of event types (A, B), the adversary has knowledge about the true mean value $\mu((A, B))$ and standard deviation $\sigma((A, B))$ of the inter-arrival time distribution of pairs of event types. Then similar to the calculation of *SI*, the *Reorder Indicator (RI)* is calculated as $1 - p_z(A_i, B_j)$ where $p_z(A_i, B_j)$ is calculated from the corresponding z-score. If RI is above a pre-defined threshold, then the adversary assumes that the event pair has been reordered.

5.3 QoS Preservation

QoS Preservation for CDA: We begin our evaluation by evaluating the negative impact of concealing private patterns onto QoS. As performance metric, we use the utility metric defined in Eq. 1. We configure the utility metric such that a) no private patterns are revealed (privacy takes strict precedence over QoS i.e., $c_p = 1$); b) all public patterns are assigned uniformly random weights between 1 and 10 to express the impact of different types of public patterns onto QoS; c) consequently false positive and false negative public patterns get same weights in negative. We first evaluate the capability of our approach to conceal private patterns for different combination of pattern types (sequence, conjunction and negation). For this evaluation we use the CDA obfuscation strategy. Figure 2a show the results from both the datasets for all combinations. It can be seen that the impact of our event obfuscation approach onto utility (QoS) is very small. The utility value shown here is represented as percentage of maximum theoretical utility (utility considering only public patterns). The number of public patterns for the msnbc dataset is lesser compared to e-commerce dataset which in turn results in lesser false positives and false negatives and hence better utility.

Privacy-QoS trade-off: With increase in number of public patterns, there is a higher chance that an obfuscation to conceal a private pattern might also impact public patterns. In this respect we show the impact of our event obfuscation approaches with increase in the number of public patterns. Figure 2b shows the result for increasing number of public patterns with the number of private patterns unchanged for both the datasets. It can be seen that the number of public patterns indeed influence QoS. Besides we show the impact of increasing public patterns on the number of false positives and false negatives with the same setup for the two datasets. in Fig. 2c.

CPA Approach: The pseudo-obfuscations introduced by our CPA approach might affect some public patterns and hence there is a drop in utility at the expense

of making our obfuscations indistinguishable. Here we compare the drop in utility with and without pseudo-obfuscations. We again use the setting such that the approaches do not reveal any private patterns. To evaluate the reduction in confidence achieved by our CPA approach, the required reduction in confidence $\rho - \gamma$ is set to 20%. Also we introduce pseudo-obfuscations only in windows that do not have any private pattern matches. Now for CPA approach, we calculate κ (cf. Sect. 4), generate 10 pseudo-private patterns for each operator type, distribute these 10 pseudo patterns κ times uniformly over the selected windows. Figure 2d shows that the utility comparison between CDA and CPA obfuscation strategies with decrease in size of selected windows for introducing pseudo-event reorderings for the e-commerce database and it can be seen that the drop in utility is negligible to achieve reduction in confidence of the adversary.

Robustness to probabilistic attacks: The confidence of the adversary is evaluated by his *precision* (correctly identified private patterns divided by the overall number of correctly or incorrectly identified private patterns). Figure 2e shows the achieved reduction in adversary precision using CPA approach compared to CDA approach with decrease in the number of selected windows for pseudo-obfuscations in case of reordering for both the datasets. We select those windows with no private patterns to introduce pseudo-private patterns for this evaluation. It can be seen from the figure that, if the number of selected windows for pseudo-obfuscations decreases, then the achieved reduction in confidence of adversary also decreases. Figure 2f and Fig. 2g shows the drop in adversary precision achieved using the CPA approach with the same setup with respect to suppression and introduction of fake events respectively for both datasets.

Till now, we only evaluated with a setup where privacy took precedence over QoS, i.e., with 0% revealed private patterns. Now, we evaluate the privacy-QoS trade-off. In order to realize this trade-off we tune the criticality percentage cp_k in Equ. 5. Figure 2h and Fig. 2i shows number of private patterns and number of false positives and false negatives over cp_k respectively. The figures show that increase in values of cp increase privacy while decreasing QoS.

6 Summary and Future Works

In this paper, we proposed a hybrid pattern-level access control component for three most commonly used CEP operators namely sequence, conjunction and negation. The approach conceals private patterns using three obfuscation strategies: event reordering, event suppression and introduction of fake events. The approach besides protecting privacy by concealing private patterns, also maximizes quality of service by preserving as many public patterns as possible. We presented two approaches that maximize utility while protecting against deterministic and probabilistic adversaries. For future work, the approach could be extended to include privacy protection for other CEP operators.

References

1. Cadez, I.V., Heckerman, D., Meek, C., Smyth, P., White, S.: Visualization of navigation patterns on a web site using model-based clustering. In: KDD (2000)
2. Cao, J., Carminati, B., Ferrari, E., Tan, K.L.: ACStream: enforcing access control over data streams. In: 2009 IEEE 25th International Conference on Data Engineering (2009)
3. Chen, D., Sain, S.L., Guo, K.: Data mining for the online retail industry: a case study of RFM model-based customer segmentation using data mining. J. Database Mark. Customer Strategy Manage. **19**(3), 197–208 (2012)
4. Chen, Y., Machanavajjhala, A., Hay, M., Miklau, G.: Pegasus: data-adaptive differentially private stream processing. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, ACM (2017)
5. Cugola, G., Margara, A.: Processing flows of information. ACM Comput. Surv. **44**(3), 1–62 (2012)
6. Cugola, G., Margara, A.: Deployment strategies for distributed complex event processing. Computing **95**(2), 129–156 (2013). https://doi.org/10.1007/s00607-012-0217-9
7. He, Y., Barman, S., Wang, D., Naughton, J.F.: On the complexity of privacy-preserving complex event processing. In: Proceedings of the 30th symposium on Principles of database systems of data - PODS 2011. ACM Press (2011)
8. Kellaris, G., Papadopoulos, S., Xiao, X., Papadias, D.: Differentially private event sequences over infinite streams. Proc. VLDB Endowment **7**(12), 1155–1166 (2014)
9. Kreyzig, E.: Advanced Engineering Mathematics, 4th edn. Wiley, New York (1979)
10. Lindner, W., Meier, J.: Securing the borealis data stream engine. In: 2006 10th International Database Engineering and Applications Symposium (IDEAS 2006) (2006)
11. Luckham, D.: The power of events: an introduction to complex event processing in distributed enterprise systems. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2008. LNCS, vol. 5321, pp. 3–3. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88808-6_2
12. Mindermann, K., Riedel, F., Abdulkhaleq, A., Stach, C., Wagner, S.: Exploratory study of the privacy extension for system theoretic process analysis (STPA-Priv) to elicit privacy risks in eHealth. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (September 2017)
13. Palanisamy, S.M., Dürr, F., Tariq, M.A., Rothermel, K.: Preserving privacy and quality of service in complex event processing through event reordering. In: Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems - DEBS 2018, ACM Press (2018)
14. Petrov, C.: 25+ impressive big data statistics for 2020 (July 2020). https://techjury.net/blog/big-data-statistics/#gref
15. Quoc, D.L., Beck, M., Bhatotia, P., Chen, R., Fetzer, C., Strufe, T.: Privacy preserving stream analytics: The marriage of randomized response and approximate computing. CoRR abs/1701.05403 (2017)
16. Schilling, B., Koldehofe, B., Rothermel, K., Ramachandran, U.: Access policy consolidation for event processing systems. In: 2013 Conference on Networked Systems. IEEE (March 2013)
17. der Spek, S.V., Schaick, J.V., Bois, P.D., Haan, R.D.: Sensing human activity: GPS tracking. Sensors **9**(4), 3033–3055 (2009). https://doi.org/10.3390/s90403033
18. Wang, D., He, Y., Rundensteiner, E., Naughton, J.F.: Utility-maximizing event stream suppression. In: Proceedings of the 2013 International Conference on Management of data - SIGMOD 2013, ACM Press (2013)