# APP-CEP: Adaptive Pattern-Level Privacy Protection in Complex Event Processing Systems

Majid Lotfian Delouee[1][a], Victoria Degeler[2][b], Peter Amthor[3][c] and Boris Koldehofe[3][d]

[1]*Bernoulli Institute, University of Groningen, Groningen, Netherlands*

[2]*Informatics Institute, University of Amsterdam, Amsterdam, Netherlands*

[3]*Department of Computer Science and Automation, Technische Universität Ilmenau, Ilmenau, Germany*

*fi*

Keywords: Distributed Complex Event Processing, Stream Processing, Privacy, Pattern, Adaptation.

Abstract: Although privacy-preserving mechanisms endeavor to safeguard sensitive information at the attribute level, detected event patterns can still disclose privacy-sensitive knowledge in distributed complex event processing systems (DCEP). Events might not be inherently sensitive, but their aggregation into a pattern could still breach privacy. In this paper, we study in the context of APP-CEP the problem of integrating pattern-level privacy in event-based systems by selective assignment of obfuscation techniques to conceal private information. Compared to state-of-the-art techniques, we seek to enforce privacy independent of the actual events in streams. To support this, we acquire queries and privacy requirements using CEP-like patterns. The protection of privacy is accomplished through generating pattern dependency graphs, leading to dynamically appointing those techniques that have no consequences on detecting other sensitive patterns, as well as non-sensitive patterns required to provide acceptable Quality of Service. Besides, we model the knowledge that might be possessed by potential adversaries to violate privacy and its impacts on the obfuscation procedure. We assessed the performance of APP-CEP in a real-world scenario involving an online retailer's transactions. Our evaluation results demonstrate that APP-CEP successfully provides a privacy-utility trade-off. Modeling the background knowledge also effectively prevents adversaries from realizing the modifications in the input streams.

## 1 INTRODUCTION

Interpreting the individual's data to generate insightful information has attracted interest in various application fields such as e-commerce, public healthcare, and the Internet of Things (IoT). Such applications typically involve distributed, timely processing of data for preventive or predictive use, with constraints ranging from mere least-latency up to real-time requirements (Lotfian Delouee et al., 2022). One of the state-of-the-art paradigms for analyzing data in a distributed manner in real-time is Distributed Complex Event Processing (DCEP): Streams of simple events (e.g., IoT raw data) are analyzed and transformed into complex events representing situations of interest (e.g., queries over sensor data streams). This transformation is performed using a set of processing logic called CEP rules (Lotfian Delouee et al., 2023a). For example, a traffic monitoring system can infer a *road congestion* via simple events: $Average\_Vehicles\_Speed < 20$ km/h and $Vehicles\_Density > Normal\_Density$.

A key challenge in the analysis of such information is privacy. It leads to a conflict of goals: On the one hand, users of a DCEP system should be provided an optimal Quality of Service (QoS). As an example from the e-commerce domain, product managers should be able to detect and reason about varying sales. On the other hand, disclosing specific events (e.g., purchase details) might violate the privacy of data owners (e.g., customers of a webshop) (Lotfian Delouee et al., 2023b). Similar examples can be found for medical data of hospital patients or health insurance clients (Palanisamy et al., 2018). They share an honest-but-curious type of adversary, represented by users as well as nodes of a distributed CEP middleware.

Hence, DCEP requires *Privacy Preserving Mechanisms (PPM)* to protect the privacy of data owners.

[a] https://orcid.org/0000-0003-1326-7540
[b] https://orcid.org/0000-0001-7054-3770
[c] https://orcid.org/0000-0001-7711-4450
[d] https://orcid.org/0000-0002-1588-2056

Lotfian Delouee, M., Degeler, V., Amthor, P. and Koldehofe, B.
APP-CEP: Adaptive Pattern-Level Privacy Protection in Complex Event Processing Systems.
DOI: 10.5220/0012358700003648
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 486-497
ISBN: 978-989-758-683-5; ISSN: 2184-4356
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

Despite most PPMs (e.g., access control) operating on the level of single events (i.e., by protecting event attributes), privacy requirements are often represented by a combination of events through complex event patterns. A pattern is the CEP representation of a complex event with a set of one or more operations over simple events (e.g., filtering special events) to detect a situation. For instance, a *sequence* pattern equals the occurrence of specific events in a predefined order over single or joint streams, such as a purchase of a pregnancy test followed by children's toys.

Those privacy-sensitive patterns that data owners want to conceal are called *private patterns*. Conversely, *public patterns* are non-privacy-sensitive patterns that must be detected accurately and timely to deliver the promised services, e.g., query results. Regarding the accuracy of complex event detection, failing to detect a complex event that occurred in the real world (i.e., False Negative, FN) and false detection of events that did not actually happen (i.e., False Positive, FP) are the main QoS metrics by which the performance of a DCEP system can be evaluated.

In this work, we demonstrate that a feasible tradeoff between concealing private patterns and detecting public patterns cannot be comprehensively provided by a single, statically applied PPM. To this end, our approach is based on a dynamic assignment of different obfuscation techniques (OT) to optimize the following constraints.

First, complex event patterns share causal dependencies. This forbids a naïve solution based on just one OT, e.g., event reordering, which fails in case the reordered event stream cannot allow for any meaningful public pattern detection. Second, an adversary's background knowledge must be taken into account when choosing an OT. For example, for any event $e_1$ dropped in the result of one query, but not of another, an adversary could possibly infer parts of the original stream based on the combined knowledge of both. Third, DCEP systems are by nature dynamic: Public and private patterns can be dynamic due to changes in active queries or the data owners' privacy requirements. Input stream sources might vary spontaneously. Finally, an adversary's background knowledge monotonically increases over time.

In this paper, we present APP-CEP as a solution to these challenges. Our contributions are as follows:

1. A PPM that obtains privacy requirements and situations of interest in the form of event patterns as input and specifies the most efficient OT to conceal each private pattern when delivering the results to the queries.
2. OT selection in our proposed PPM based on a graph structure, used to model pattern dependencies and extract the input stream's features to predict the switching time between OTs to avoid privacy violations and meet scalability requirements.
3. A structure for data owners to dynamically model the adversary's potential background knowledge based on an event dependency set and possible statistical event-related information gathered from the available event streams' history.
4. Performance evaluation based on a real-world dataset to show the ability of APP-CEP to boost the performance of the DCEP systems in real-world scenarios.

The remainder of this paper is structured as follows. We further detail the problem, particularly for an online shop scenario, and motivate the need for pattern-level privacy in DCEP systems in Section 2. We introduce an overview of the APP-CEP system model in Section 3. We formalize the problem statement in Section 4. Section 5 presents the detailed overview of APP-CEP. The evaluation results of APP-CEP are exhibited in Section 6. The related work is presented in Section 7. Finally, Section 8 concludes our paper and points to our future work.

## 2 CASE STUDY: WEBSHOP

In a recent project by the National Statistical Institute of Norway (Gundersen, 2022), several grocery chains have been ordered to share all their receipt data with this statistics agency. To protect privacy, this institute claims that by performing *pseudonymization*, the account number, which can identify a person, will be changed to another unique number (i.e., attribute-level access control). However, pattern-level correlation plus background knowledge would enable the adversaries to realize customers' identities, violating their privacy.

In this section, we introduce a scenario based on the transaction set of an online retailer that demonstrates the application of APP-CEP in real-world situations similar to the mentioned Norwegian example (see Figure 1). In such a scenario, a state-of-the-art PPM fails to provide an acceptable level of privacy protection since they mainly protect at the level of events (e.g., by controlling the access using event attributes obfuscation). We consider all transactions of a retailer selling all-occasion gifts as the input stream and attempt to detect predefined public and private patterns. Any user (e.g., webshop's manager) can issue a query to detect situations of interest (e.g., top-selling products in a particular period) that are active for a specific time. On the other hand, data owners are

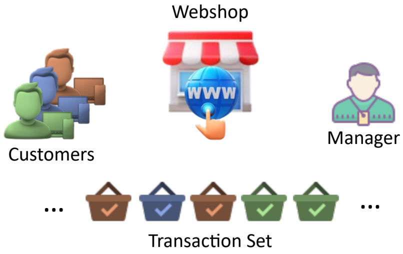


Figure 1: Webshop Scenario. The system aims to protect customers' privacy while answering webshop manager queries.

customers of the webshop, and each of their purchases triggers an event in this scenario (i.e., using their customer ID) and is labeled with a timestamp. Lining up the purchase records according to their timestamp generates a transaction set event stream, which is analyzed to detect valuable real-time correlations.

By employing APP-CEP, a customer can specify the privacy requirements, e.g., concealing a Christmas party. In this scenario, we assume the sensitive information can be represented by CEP patterns (e.g., sequence, conjunction, negation, etc.). For instance, a Christmas party can be represented by *CONJ (Invitation Card, Christmas Ribbon, Christmas Decorations)*. To conceal such sensitive information, the chosen obfuscation technique (e.g., drop event for *Invitation Cards*) might influence the results of non-sensitive queries (e.g., top-selling products). Moreover, various parameters (e.g., pattern dependencies, event distribution in the input stream, etc.) should be considered carefully. Hence, the goal of APP-CEP is to answer user queries intelligently in such a way that the privacy of customers is not violated.

# 3 SYSTEM MODEL

This section presents the components of APP-CEP, the model of event sources, and obfuscation models.

## 3.1 APP-CEP Model

We consider a typical DCEP system with multiple *producers* (e.g., IoT sensors) that advertise the simple event streams they can provide. Each of these streams is related to one or more *Data Owners* (DO), meaning the provided data belongs to this data owner(s) from a privacy protection point of view. For example, the location event stream of a car belongs to its driver. Correspondingly, *consumers* (e.g., applications, services, etc.) submit their situations of interest as continuous queries to the system. Hence, the set $Q = \{q_1, ..., q_n\}$ denotes the set of running queries that must be responded to by processing the events of current input streams. In addition, a set of *brokers* (e.g., CEP engines) perform event processing tasks (e.g., drop, reorder, union, tamper, etc.) by hosting corresponding *operators* and delivering the resulting stream(s) to the next step.

A query $q_i$ determines the logic to detect complex events by applying standard CEP operators over simple events' attributes (e.g., pattern matching). To this end, each detection logic needs to be hosted by a specific operator to be executed. Moreover, each data owner is able to describe its privacy requirements and deliver them to the system. Such requirements should be accompanied by a normalized weight that shows their importance. Hence, the set

$$PR(DO_i) = \{(pr_1, w_{pr_1}), ..., (pr_m, w_{pr_m})\}$$

shows the privacy requirements specified by data owner $DO_i$ along with their corresponding weights. As another query feature, the importance of queries might vary due to their priorities. That is why detecting such query patterns is of more significance to the system instead of concealing privacy requirements, e.g., life-related queries in a healthcare scenario.

## 3.2 Event Source Model

Producers, also called *event sources*, are the origin of the events that generate simple event streams. As an example, if we implement our mechanism in an IoT scenario, the sensors that measure a specific phenomenon (e.g., location of a target) act as event sources. Regarding the number of targets that can be covered, we categorized the event sources into two groups: *DO-Specific* and *Multi-DO*. For the former, the event source can only be used for queries related to the corresponding data owner. In other words, the generated stream will not reveal privacy-sensitive information about other targets (i.e., data owner). For instance, a cellphone's GPS signal can only be used for tracking a specific target. For the latter, information about multiple targets can be acquired by analyzing such event source streams. For instance, a stationary security camera delivers data related to various targets. These event sources should be used cautiously in any privacy-aware data analysis approaches.

In addition, the event sources here are interconnected to the system over a wireless network and must be registered in the system due to privacy concerns. Besides, each data owner can act as a CEP consumer by submitting queries. CEP operators can also be hosted in nodes with sufficient computing capabilities, e.g., on the cloud or fog node in an IoT scenario.

### 3.3 Obfuscation Model

An obfuscation technique is a real-time modification in the order, occurrences, or values of event attributes that results in concealing sensitive information (i.e., private patterns). Such a modification can be applied to event streams, executed prior to the CEP middleware. More importantly, the obfuscation technique should be placed on the earliest available, trustworthy resources to the producers, preventing inconsistencies in the delivered streams. Among common obfuscation techniques, we consider a PPM is able to hire the following techniques:

- **Suppression.** Removing the occurrence of events by dropping them from the event stream, e.g., removing the event $HomePage\_Visit^{c_1}$ to conceal page navigation behavior of customer $c_1$.
- **Reordering.** Changing the order of two events by swapping their timestamps, e.g., changing the occurrence time of an event $Buy_i^{c_1}$ with the event $Buy_j^{c_1}$ to conceal the order of purchases for a specific customer $c_1$.
- **Injection.** Introducing fake events and inserting them in unique places in the stream, e.g., injecting the event $Buy_j^{c_1}$ multiple times to conceal a specific interest to item $i$. It shows the person $c_1$ interested in both items, i.e., $i$ and $j$.
- **Tampering.** Changing attributes of an event to a wrong value or noise injection in event timestamps, e.g., changing the value of item type $i$ in $Add\_To\_Basket_i^{c_1}$ to $j$ which produces $Add\_To\_Basket_j^{c_1}$ in order to conceal unhealthily shopping habit of a diabetic customer from an insurance company.
- **Generalization.** Anonymizing the event's attributes by changing to a general value, e.g., generalizing an event $Buy_{apple}^{c_2}$ to $Buy_{fruit}^{c_2}$ to conceal a customer's shopping interests.
- **Hybrid.** A combination of thereof, e.g., suppression of event $Buy_{medicine}^{c_2}$ and injecting event $Buy_{drink}^{c_2}$ to conceal a customer's decease.

## 4 PROBLEM STATEMENT

Consider as inputs of a stream processing system sets of public and private patterns. Basically, a PPM requires checking the actual events in the input stream to compare and assign OTs on the fly (i.e., by limiting the stream dimension using windowing) (Palanisamy, 2020). Therefore, adaptive decisions for maintaining the privacy-QoS trade-off should be performed at run-time by exchanging between the best possible PPMs. Such run-time changes are also referred to as *transitions* (Alt et al., 2019). Performing transitions between PPMs requires careful treatment regarding the required time and resources. Moreover, transitions can impose gaps during which PPM cannot meet the privacy requirements (i.e., periods with no applied obfuscation). Also, it will lead to an oscillation between obfuscation techniques that will worsen by scaling up the pattern sets since it will drastically increase the number of switches. Consequently, finding a solution to predict transition points is necessary to minimize the overhead and prevent oscillation.

Moreover, the adversary's background knowledge level is determined in such systems in the initial stage. However, an *Omnipresent adversary* can increase its knowledge by issuing more and more queries and trying to correlate the generated results. This indicates another weakness of current approaches that do not consider this dynamicity. In addition, it again approves the idea of adaptive assignment of OT models to private patterns since new knowledge obtained by adversaries should be taken into account in the assignment procedure. So, a comprehensive PPM should also provide enough means for dynamic modeling of the adversary's background knowledge.

Therefore, the main problem this paper solves is obtaining data owners' privacy requirements in run-time in the form of patterns and trying to conceal them in a way that leads to maximizing the Quality of Service (QoS) (i.e., utility) while considering potential adversary's knowledge. Here, the comparison metric is calculated based on the number of truly detected matches of public patterns (i.e., $TP_{PUB}$) and False Positives (FP) in detecting public patterns (i.e., $FP_{PUB}$). In addition, regarding private patterns, the number of truly obfuscated matches of private patterns (i.e., $TO_{PRIV}$) and wrongly created new private patterns after applying obfuscation models (i.e., $FR_{PRIV}$) are being involved. Finally, the last considered metric is the number of matches for private patterns that can be derived using the adversary's background knowledge.

Each of these event types (e.g., TP) has its own group weight (e.g., $W_{TP}$). The rationale behind these coefficients is in some applications, the importance of true detection of a phenomenon is not equal to the wrong extra detection or vice versa. The same can be defined as concealing a private pattern over generating not true new private patterns. For example, in a healthcare scenario, vital situations of patients (i.e., public patterns) must be detected regardless of the individual's privacy protection (i.e., private patterns). Besides, the significance of private patterns within their set is not equal (i.e., pattern priority). To consider this fact, we define a unique weight for each pattern (e.g., $w_{to}$). The same also applies to public patterns (e.g., $w_{tp}$).

According to the mentioned event groups, a *Privacy-Utility Trade-off* (PUT) function is formulated as an optimization problem to make the performance of each OT model comparable. We extend the objective function formulation in (Palanisamy, 2020) i) to deal with the dynamicity of matched private and public patterns, ii) and quantify the adversary's background knowledge.

More formally, the resulting formula is:

$$
\begin{aligned}
Max \quad & \Bigg\{ \Bigg( W_{TP} \sum_{tp \in TP_{PUB}} w_{tp} \Bigg) - \Bigg( W_{FP} \sum_{fp \in FP_{PUB}} w_{fp} \Bigg) \\
& + \Bigg( W_{TO} \sum_{to \in TO_{PRIV}} w_{to} \Bigg) - \Bigg( W_{FR} \sum_{fr \in FR_{PRIV}} w_{fr} \Bigg) \\
& - \Bigg( W_{Adv} \sum_{p \in PRIV} w_{p} \Bigg) \Bigg\}
\end{aligned}
$$

This objective function generally can be used in any application context. In other words, the last part is calculated by estimating the number of already obfuscated private pattern matches, which can be revealed based on the adversary's background knowledge. However, if such a piece of knowledge is not available enough, the last part of this objective function can be skipped to make a fair comparison.

# 5 THE APP-CEP DESIGN

In Figure 2, we illustrate the main functionalities of APP-CEP. Parts of our system are built on existing concepts for converting the privacy requirements and situations of interest in natural language to a CEP-like pattern representation (Stach and Steimle, 2019). Therefore, we assume that data owners and users are able to feed the system with public and private patterns. The main goal of APP-CEP is to gather related information about ① patterns' dependencies, ② event dependency set, as well as ③ input stream features to opt for ④ obfuscation techniques which satisfy PUT, even without knowledge about the actual events in the input streams and adapt the selection based on the obfuscation results in execution time. This section will further elaborate on the functionality of these four components of APP-CEP.

Here, there are two dynamic databases, namely *stream database* and *OT database*, which assist APP-CEP in the obfuscation procedure by providing up-to-date information. In the former, all previous events are collected (e.g., the transaction history of the webshop) for further analysis, such as extracting event dependencies and stream features. In the latter, all available models of applying obfuscation techniques (e.g., dropping the first event is a private pattern) are gathered to support performance analysis of deployed obfuscation technique and make adaptation decisions. Moreover, we presume a data owner has sufficient skills to support APP-CEP by dynamically providing event dependencies to facilitate the obfuscation procedure in the execution time. For example, a webshop customer can provide information about his preferred pick-up point for purchases that will be delivered in the working days since such information should be considered while obfuscating the patterns in location tracking applications.

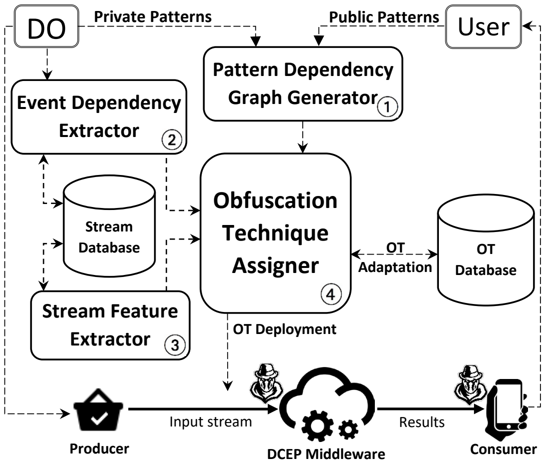


Figure 2: The APP-CEP Proposed System Design. The solid lines indicate the data/event flow, while the dashed indicates the control flow.

In preliminary approaches on multiple pattern-type privacy protection (Palanisamy, 2020) , a PPM is only able to calculate a utility value by inspecting the actual events in the input streams. Therefore, assigning the best obfuscation technique for each private pattern is achievable by comparing the utility values. However, a significant disadvantage of such strategies is the complexity of designing a low-overhead adaptation procedure to maintain the privacy protection performance at an acceptable level concerning the system's dynamics, e.g., changing data owners' privacy requirements . To appropriately respond to dynamics, a greedy idea would be to switch between obfuscation techniques whenever a model with a higher utility value is provided by the corresponding component (i.e., Obfuscation Technique Assigner). The main pitfalls with this approach are, firstly, time and resource overhead for transitions and, secondly, oscillation between obfuscation techniques that downgrade the ob-

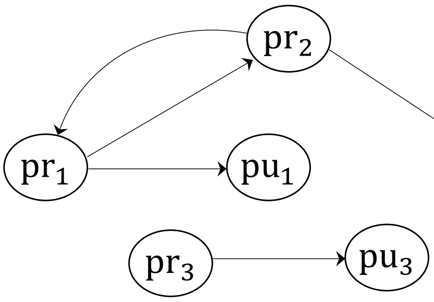


Figure 3: The global pattern dependency graph represents the impacts of obfuscating each private pattern on related public/private patterns.

fuscation performance.

To overcome the mentioned challenges, one could realize that relying only on the actual events that appear in the input streams causes transition overheads and oscillation issues. In other words, if a PPM is able to predict the obfuscation techniques' transition time, the data/operator migration can be performed ahead of time in order to minimize inefficiency in both time and resource utilization. Such prediction can be achieved by realizing the correlation between patterns as well as events. Moreover, extracting input stream features will enable a PPM to prevent unnecessary obfuscation transitions, leading to performance efficiency. For example, the number of pattern matches can roughly be estimated by extracting the event distribution in each event stream. This will prevent oscillation in obfuscation assignment, which might be produced by transition from dropping to reordering and quickly returning to dropping.

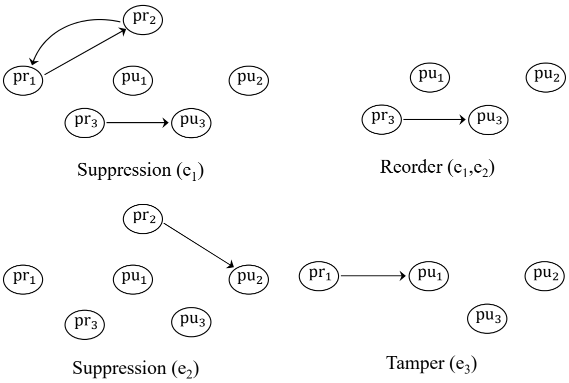
Nevertheless, generating more up-to-date insights about patterns and also events helps APP-CEP to wisely adapt the system to maximize the overall performance of the obfuscation procedure.

## 5.1 Pattern Dependency Graph

Predicting the obfuscation transition time is possible by determining the potential dependencies between private and public patterns. To be more precise, consider the following public and private patterns.

$$\begin{aligned} PUBLIC = \{\ & \{pu_1 \mid Buy^{p_i}_{CD} \rightarrow Return^{p_i}_{CD'}\}, \\ & \{pu_2 \mid Buy^{*}_{BC} \rightarrow Buy^{*}_{BC} \rightarrow Buy^{*}_{BC}\}, \\ & \{pu_3 \mid Buy^{p_i}_{AD} \mid\mid Buy^{p_i}_{FJ}\}\ \} \\ PRIVATE = \{\ & \{pr_1 \mid Buy^{p_1}_{IC} \mid\mid Buy^{p_1}_{CR} \mid\mid Buy^{p_1}_{CD}\}, \\ & \{pr_2 \mid Buy^{p_1}_{IC} \mid\mid Buy^{p_1}_{BC}\}, \\ & \{pr_3 \mid Buy^{p_1}_{AD} \rightarrow Buy^{p_1}_{DM}\}\ \} \end{aligned}$$

These sample pattern sets indicate the following event correlations:



Suppression (e$_1$)

Reorder (e$_1$,e$_2$)

Suppression (e$_2$)

Tamper (e$_3$)

Figure 4: The alternative pattern dependency sub-graphs represent the obfuscation impacts based on enforcing a specific obfuscation technique.

- $pu_1$. A causal relationship between buying and returning two types of Christmas decorations (CD) with the same customer ID, which shows the customers' interests.
- $pu_2$. Multiple purchases of the same product (i.e., birthday candle (BC)) by different customers, which helps recommendation systems.
- $pu_3$. The concurrent ordering of alcoholic drinks (AD) and fruit juice (FJ) with the same customer ID, which detects customers' drinking interests.
- $pr_1$. Simultaneous purchasing of invitation cards (IC), Christmas ribbons (CR), and Christmas decorations, which reveals a Christmas party.
- $pr_2$. The concurrent purchase of invitation cards and birthday candles, which reveals a birthday party.
- $pr_3$. A causal ordering dependency of alcoholic drinks and diabetic medicine (DM) reveals a bad lifestyle for a diabetic person.

The respective pattern dependency graph for the mentioned patterns is depicted in Figure 3. Here, the submitted privacy requirements belonging to an individual customer (i.e., $p_1$) have been used to define private patterns. Besides, queries represented as public patterns are generally defined, not for a specific customer ID. Signs ($\rightarrow$) and ($||$) indicate the causal dependency in a sequence pattern and parallel occurrence of events in a conjunction pattern, respectively.

To construct a pattern dependency graph, we assume that nodes of the graph are patterns, and for each private pattern, outgoing edges are one-way arrows from this private pattern to all dependent public/private patterns. A pattern is called the *neighbor* of private pattern *pr* if an edge exists going from *pr* to this specific node. For instance, if we plan to obfuscate private pattern $pr_1$, suppression of event $Buy^{p_1}_{IC}$

helps to obfuscate private pattern $pr_1$ (positive impact). However, suppression of the event $Buy_{CD}^{p_1}$ will result in false negatives in the detection of public pattern $pu_1$ (negative impact).

The graph representation of pattern dependencies supports deriving insights that can be helpful in OT model selection. For example, the outgoing edges of each node indicate the detection of how many patterns will be influenced. In a particular case, disconnected nodes (i.e., nodes with no neighbor) do not have any relationship with other nodes. Hence, the obfuscation selection is much easier and even static in such cases until their connectivity changes in the updated pattern dependency graph. Any available obfuscation technique can be chosen in such situations since they all support the PUT goal.

On the other side, if a private pattern has one or more outgoing edges in the global dependency graph, an alternative solution would be to break down the global dependency graph into obfuscation-specific sub-graphs (e.g., suppression($e_1$) graph, which drops the first event of all matches). This way, APP-CEP can find a sub-graph in which this specific node has no neighbors. Figure 4 shows the generated graphs of four different obfuscation techniques as examples of sub-graphs. Here, nodes $pr_1$ and $pr_3$ are disconnected nodes in the Suppression($e_2$), and $pr_2$ has no neighbors in the Suppression($e_1$). Therefore, these OT models are potential candidates for corresponding private patterns. Note that Reorder($e_1$, $e_2$) is not among available OT models for $pr_1$ and $pr_3$ because the conjunction operator is used in the definition of those patterns between the first and second events, and such obfuscation technique is not able to conceal these patterns. Similarly, Tamper($e_3$) is not applicable for patterns of length 2, e.g., $pr_2$ and $pr_3$.

## 5.2 Event Dependencies

Modeling the adversaries' background knowledge is a complicated task that has yet to be well-studied in this field. Some research works have been conducted to statistically predict special privacy attacks to model the adversary (Palanisamy, 2020). For example, an adversary might infer several event types' true mean inter-arrival time from stream history. Then, he can realize enforced pattern obfuscation by computing statistical metrics like the suppression probability of this event type and try reconstructing the original input stream. However, the potential data owners' information is not exploited to complement the final adversary model.

To this end, APP-CEP supports the runtime expression of event dependencies. This facilitates modeling the possible information that an adversary might use to derive insights related to data owners. For instance, the regular shopping habits of a customer, e.g., buying two products always together, give a hint to the system not to drop one while publishing the other. In more detail, four types of event dependencies can be introduced to be considered in the obfuscation assignment phase.

### 5.2.1 Causal Dependency ($\rightarrow$)

In this category, a relationship between two events can be specified in two ways: firstly, one event is the reason for occurrences of the other (i.e., cause and effect), and secondly, one event always happens before the other. For example, in our use-case scenario, a causal dependency can be derived from history as:
$CD\ (Submit\_Order_i^{c_1}\ , Payment\_Successful_i^{c_1})$
which means the event of submitting an order is the cause, and the successful payment event is the effect. Also, a customer $c_1$ might express that he always checks the available balance before adding a product $p$ to the shopping basket. Therefore, the customer $c_1$ can provide a causal dependency to the system as:
$CD\ (Check\_Balance^{c_1}\ , Add\_To\_Basket_i^{c_1})$

In these cases, APP-CEP considers the dependency, e.g., does not suppress the cause event and keeps the effect event or reorders these two events.

### 5.2.2 Parallel Occurrence ( $||$ )

Special events (two or more) happen always, or at least with a high probability, simultaneously or in a short period. This means an intelligent obfuscation assignment should consider them as a *single combined event*, e.g., reorder all events instead of one. As an example in the proposed webshop scenario, a customer $c_1$ might express that he always buys products $p$ and $q$ together. This leads to a parallel occurrence of the corresponding two events, represented as:
$PO\ (Add\_To\_Basket_p^{c_1}\ , Add\_To\_Basket_q^{c_1})$.

### 5.2.3 Infeasible Events ($\nrightarrow$)

Another type of event relationship also shows the impossible occurrences of one or several events or patterns. For instance, when a customer $c_1$ is navigated to the bank payment page as a result of the *Submit_Order* event with id $i$, it is not possible to observe two events of *Payment_Successful* and *Payment_Rejected* afterward for this special order submission in the stream, that can be expressed by:
$IE\ [CD\ (Submit\_Order_i^{c_1}\ , Payment\_Successful_i^{c_1}),$
$CD\ (Submit\_Order_i^{c_1}\ , Payment\_Rejected_i^{c_1})]$

Such dependencies usually cannot be expressed by a data owner, but require help from domain experts in the application scenario.

### 5.2.4 Periodic Events ($\bar{X}$)

Apart from dependencies between events, this type of information shows insights can be derived as a result of statistical calculation over a specific event type for a unique data owner. We assume that the adversary has the required computing resources and access to the data history to correlate the statistical dependencies for a particular data owner. For example, a customer $c_1$ might have a shopping habit of buying a special product $p$ each day. Hence, such a habit can be expressed by:
$PE\ (Buy_p^{c_1}\ , day\ )$
Therefore, applying any obfuscation technique related to dropping or altering such events (e.g., suppression, tampering, generalization, etc.) will lead to violating the privacy of data owner by revealing the stream modification on the adversary side, whereas obfuscation techniques like reordering still might be applicable in such situations.

## 5.3 Stream Features

Even if we know the privacy requirements of data owners and the dependencies between patterns and events, the input streams also bring up another dynamicity that should be considered. In more detail, the number of matches for a pattern determines another vital dimension. For example, a Christmas party is more likely to be revealed during Christmas. Therefore, obfuscating a private pattern that increases the reveal of any Christmas party at another time of the year (e.g., in July) would be acceptable because the detection probability of such a party is very low in other periods.

Hence, extracting input stream features helps a PPM to assign the OT model more efficiently. To do so, APP-CEP investigates the stream database and analyzes the input streams online to adapt the features. As mentioned earlier, one significant feature of streams is the average number of matches for each pattern. For instance, the number of matches for a Christmas party pattern is calculated weekly, which helps dynamically assign weight to this private pattern in the PUT function.

Apart from patterns, event-related stream features bring up the benefits, causing a more brilliant OT model selection. For instance, the event distribution helps estimate a more accurate weight for each suppression model of a private pattern (He et al., 2011). If the event $e$ on a private pattern $pr$ has a higher frequency in history, suppression of $e$ tends to create more FN in detecting neighbors of $pr$ which contain $e$. Such an idea can also be extended to subsets of each private pattern instead of a single event by checking its distribution for other types of obfuscation (e.g., a two-event subset for reordering).

Algorithm 1: Obfuscation Technique Assignment.

```
Initialization:
    $\mathcal{PU}$ , $\mathcal{PR}$ $\leftarrow \varnothing$;
    $\mathcal{D}$ $\leftarrow$ Event Dependencies;
    $\mathcal{O}$ $\leftarrow$ Obfuscation Techniques;
upon ($Submit(Q_i)$) do
    $\mathcal{PU} \leftarrow \mathcal{PU} \cup$ CEP-Pattern($Q_i$);        ▷ Query
    OT_Assigner ($\mathcal{PU}$, $\mathcal{PR}$, $\mathcal{D}$);
upon ($Submit(Pr_i)$) do
    $\mathcal{PR} \leftarrow \mathcal{PR} \cup$ CEP-Pattern($Pr_i$);   ▷ Priv Req
    OT_Assigner ($\mathcal{PU}$, $\mathcal{PR}$, $\mathcal{D}$);
upon ($Submit(d_i)$) do
    $\mathcal{D} \leftarrow \mathcal{D} \cup d_i$;               ▷ Event Dependency
    OT_Assigner ($\mathcal{PU}$, $\mathcal{PR}$, $\mathcal{D}$);
function OT_ASSIGNER($\mathcal{PU}$, $\mathcal{PR}$, $\mathcal{D}$)
    $\mathcal{G}$ $\leftarrow$ Pattern Dependency Graphs;
    $\mathcal{F}$ $\leftarrow$ Input Stream Features;
    for $pr \in \mathcal{PR}$ do
        $\mathcal{O}^{pr} \leftarrow \mathcal{O}$;
        for $ot \in \mathcal{O}$ do
            if $ot$ violates $\mathcal{D}|\mathcal{F}$ then
                $\mathcal{O}^{pr} \leftarrow \mathcal{O}^{pr} - ot$ ;
        $\mathcal{OT}^{pr} \leftarrow \varnothing$;
        for $ot \in \mathcal{O}^{pr}$ do
            if $OutDeg_{g_{ot}}^{pr}$ = 0 then
                $\mathcal{OT}^{pr} \leftarrow \mathcal{OT}^{pr} \cup ot$;
        if $\mathcal{OT}^{pr} \neq \varnothing$ then
            $\alpha_{pr} \leftarrow \mathcal{OT}_1^{pr}$;
        else
            $\alpha_{pr} \leftarrow Random(\mathcal{O}^{pr})$;
    $\alpha \leftarrow \{(pr_1, \alpha_{pr_1}), ..., (pr_n, \alpha_{pr_n})\}$; ▷ Solution
```

## 5.4 Obfuscation Technique Assignment

In connection with the OT database, the *obfuscation technique assigner* module, which is the core component of APP-CEP, acts as the system's brain. It aggregates the information collected from the other three modules (i.e., ① pattern dependency graph generator, ② event dependency extractor, and ③ stream feature extractor) to decide which OT model should be chosen regardless of the current input streams for each private pattern.

In Algorithm 1, the sets of public and private patterns are initialized, and the event dependencies derived from streams' history alongside the available obfuscation operators are delivered as inputs (line 1). Upon receiving new public or private patterns, APP-CEP updates the obfuscation technique assignment (lines 2-7). Here, we also accept the event dependencies the data owner enters at runtime (lines 8-10). The *OT_Assigner* function generates pattern dependency graphs for each OT model and updates them on runtime based on changes in both public and private sets. Also, it extracts input stream features based on history and updates those features using the current streams (lines 12-13).

According to the event dependency set and stream features, for each private pattern *pr*, APP-CEP performs the following tasks: first, it removes those OT models whose obfuscation can be realized by an adversary (lines 16-18). For example, if the result of reordering violates a causal dependency, such an OT model will be removed from the list of available OTs. In the remaining OT model space (i.e., $O^{pr}$), APP-CEP looks for a member whose dependency graph has a node with an out-degree equal to zero for *pr*, so-called *zero-out-degree* (ZOD) node (lines 20-22). One of the OT models containing such a node will be assigned to the *pr*. A random OT model will be assigned to a *pr* if no ZOD node has been found for this private pattern (lines 23-26). Finally, the assigned OT model for each private pattern is determined and will be updated upon any change in patterns or event dependencies.

Although APP-CEP is a restrictive approach since the idea of selecting a ZOD node does not achieve the maximum QoS, finding these nodes guarantees both obfuscating each private pattern and trying to detect as many public patterns as possible. In addition, calculating the exact PUT value requires considering the exact number of pattern matches in each input stream, whose disadvantages have been discussed earlier. In case of finding no ZOD node, we believe assigning the best OT model is outside the scope of our paper. Therefore, we randomly assign the OT model in such a worst-case scenario.

## 6 EVALUATION

The evaluation investigates the following questions:

1. Can graph-based pattern-level privacy provide a trade-off between privacy concerns and the system's promised QoS?
2. What limitations are involved in applying pattern-level privacy in real-world scenarios?

We created a single Virtual Machine (VM) in Oracle VM Virtual Box Manager, in which we installed Ubuntu version 20 OS. We allocated 6 CPU cores with 100 percent execution capacity and 24 GB of main memory to the VM. Furthermore, for event detection, we build on *FlinkCEP* (Flink, 2017), a library implemented on top of Apache Flink. We evaluated our ideas by analyzing a publicly available real-world dataset (Chen et al., 2012), transactions of an online retailer selling all-occasion gifts. A part of this dataset is selected for this evaluation, including 12000 customer purchase records. We selected the 50 most frequent patterns (length 3) from which public and private pattern sets were randomly formed. We also performed another statistical analysis to extract the most parallel purchased items and periodically purchased items as dependencies considered for the adversary's background knowledge. We executed the event detection process ten times for each combination of public and private patterns and stated the average results.

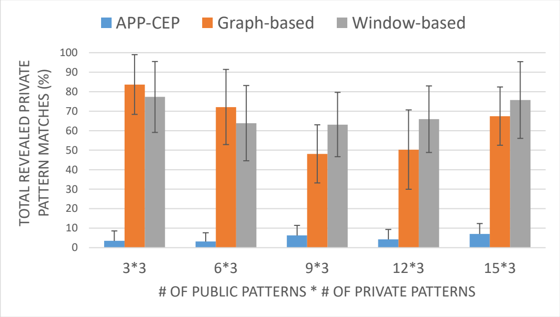


Figure 5: Percentage of total revealed private pattern matches Vs. The increasing number of public patterns.

Since modeling the adversary's background knowledge and graph-based OT assignment have not been addressed before, we selected two baseline approaches to analyze the performance of APP-CEP. The first technique, called *Window-based*, simulates the performance of the state-of-the-art mechanisms (Palanisamy, 2020; Palanisamy et al., 2018), which relies on splitting the input stream into multiple finite windows of events. This approach tries to select an obfuscation technique for each private pattern match precisely based on a utility value calculated according to the expected number of matches for all patterns after applying that specific obfuscation technique. The second approach, called *Graph-based*, performs the obfuscation technique selection similar to APP-CEP but without considering the adversary's background knowledge (i.e., event dependencies). It only checks the pattern dependency graph or OT-specific sub-graphs, if needed, and makes a decision accordingly.

Figure 5 indicates the total percentage of revealed private pattern matches (i.e., the summation of matches for data owners' privacy requirements added

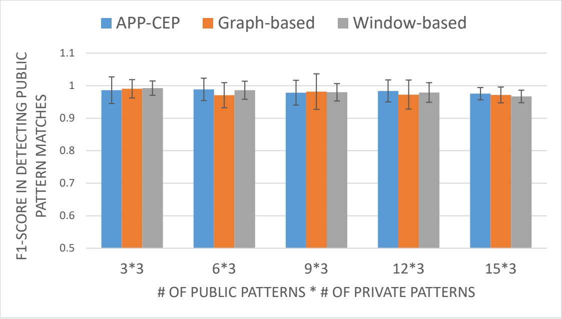


Figure 6: F1-score in detecting public pattern matches Vs. The increasing number of public patterns.

to the number of matches revealed by the adversary's background knowledge, totally divided by the actual matches) while the number of public patterns is increasing. Error bars also indicate the standard deviation of ten rounds of simulations for each approach. It can be seen that modeling and considering the adversary's background knowledge helps APP-CEP to significantly reveal less (i.e., conceal more) private patterns than other approaches. By adding more queries to the system, Graph-based and Window-based techniques fluctuate, but APP-CEP's results stay around ten percent. The random assignment of obfuscation techniques when APP-CEP cannot find a ZOD node in the global dependency graph or in obfuscation-specific sub-graphs shows its impacts in the last three bars. That is why complementing our algorithm with an intelligent solution for such situations would be challenging but necessary for our future work to solve the potential scalability issue.

Similar to FN and FP, *F1-score* is a well-known performance measure in machine learning approaches. It combines the other two measures, *precision* and *recall*, which are mainly employed to distinguish between classifiers (Li et al., 2020) in terms of accuracy. Therefore, the F1-score can be used in stream processing to compare mechanisms in terms of accuracy. From the public pattern detection's accuracy point of view, by increasing the number of involved queries, all three approaches fluctuate but gradually degrade; thereby, there is not a specific front-runner, illustrated in Figure 6. However, APP-CEP's detection accuracy is mostly equal to or better than the Window-based approach when increasing the number of involved queries (i.e., public patterns), which confirms a slight improvement over the literature, even in this challenging comparison.

The effects of increasing the knowledge of the adversary are depicted in Figure 7. It can be seen that APP-CEP reveals significantly fewer matches than the other two approaches. By involving more dependencies in query processing, the percentage of expected revealed private pattern matches unexpectedly rises slightly but is still negligible compared to Graph-based and Window-based approaches. Here, the Graph-based approach reveals little more private pattern matches than the Window-based, but generally, both approaches show similar performance. Adding more dependencies corresponds to a more knowledgeable adversary that even impacts the performance of APP-CEP. Protecting against such an adversary is almost impossible without sacrificing utility in the system. However, the proposed approach shows such a trade-off can be achievable even independently of the actual events in the stream. Moreover, note that it becomes hard to compensate for the revealed matches without modeling the adversary's additional background knowledge.

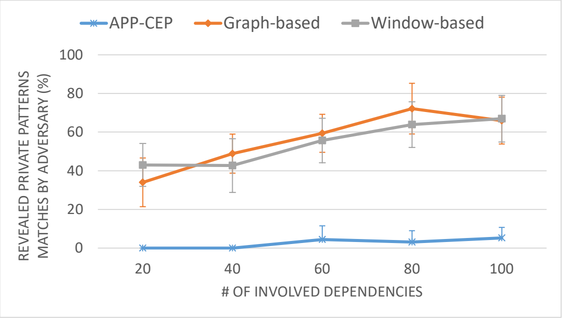


Figure 7: Percentage of revealed private pattern matches by adversary Vs. The increasing adversary's background knowledge (number of dependencies).

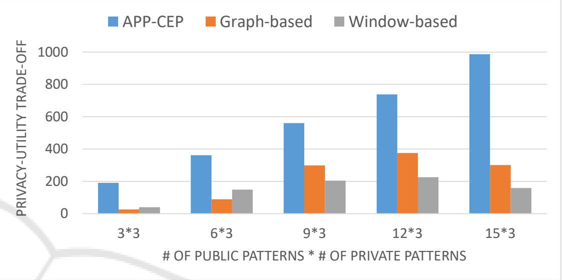


Figure 8: Privacy-Utility Trade-off Vs. The increasing number of public patterns.

Last, Figure 8 accumulates the previous results to compare the privacy-utility trade-off of the three approaches. As discussed in Section 4, the main goal of this paper is to provide a method to maximize the trade-off value (cf. PUT equation).

The computation of such a value requires the system designer to opt for an optimized value for weights. We decided to set equal weights for revealed private pattern matches and expected revealed matches by the adversary. This weight is the ratio of the average of actual public matches to the average of actual private matches for each pattern set combination. For instance, the weight for the scenario in which we have 3 public patterns and 3 private patterns was calculated as 1.0301. Also, we set the weight for the detected public pattern matches equal to 1.

Here, the results indicate that APP-CEP is capable of achieving the expected PUT even by scaling up the system's involved queries. Although the Window-based approach had a good start, the bars fall once more public pattern matches are going to be detected, as the amount of revealed private matches by the adversary significantly impacts the PUT value. The Graph-based approach seemingly had a better performance than the Window-base. However, its results also had a downward ending for the same reason.

Ultimately, the evaluation results prove the expected improvements provided by graph-based pattern-level privacy protection, solving the literature's PUT gap. APP-CEP performs better in almost all aspects discussed in this section than approaches presented in previous studies due to considering both pattern dependencies and modeling the adversary's background knowledge.

## 7 RELATED WORK

Although sharing data between stakeholders increases the utility of IoT applications, ensuring efficient preservation of privacy is still challenging (Tran et al., 2023). Similarly, providing privacy in DCEP systems requires establishing a trade-off concerning QoS since protecting an individual's privacy should not sacrifice the application's functionality, which lays on more data to maximize the quality (Plagemann et al., 2022; Schilling et al., 2013). Therefore, PPMs are required to be lightweight and do not prevent any application from detecting query patterns adequately. Moreover, although data owners have been recently becoming more aware of how their data will be analyzed and being able to control it (Tokas et al., 2023; Leicht and Heisel, 2023; Barber and Furnell, 2022), providing sufficient means for them to express their privacy concerns is still poorly studied.

The initial idea of preserving the individual's privacy by employing patterns was introduced more than a decade ago (He et al., 2011). It was theoretically proved that by employing a probabilistic model, the choice of event suppression can be estimated to maximize the utility alongside a window-based approach, which makes the obfuscation decision according to the actual events. Similar methods computed the event distribution using advanced pattern match cardinality estimation techniques (Wang et al., 2013). Other obfuscation operators (e.g., Reordering) have been discovered in the literature as an alternative to suppression operator (Palanisamy et al., 2018). The main drawback of the mentioned studies is they failed to obfuscate various pattern types since they focused only on sequence type. In addition, they were highly dependent on the input streams to apply obfuscation techniques, i.e., they failed to obfuscate various kinds of private patterns well in an environment with dynamic input event streams. Moreover, they have not supported simultaneous obfuscation techniques customized for each private pattern.

Recently, the idea of multi-operator multi-pattern privacy protection has been introduced (Palanisamy, 2020). Although this ILP-based approach maximizes the QoS while preserving privacy, it is limited to three pattern types and three obfuscation operators. Besides, it suffers from oscillation between OT models since they change the deployment as soon as they detect a change in the input stream. This instability in the deployed OT models degrades performance by imposing unnecessary transition overhead. Besides, the adversary's background knowledge, which might be used to realize the obfuscation, needed to be considered thoroughly.

## 8 CONCLUSION

In this work, we proposed APP-CEP, which represents how to enable dynamic integration of pattern-level privacy in event-based systems to protect privacy while providing an acceptable level of QoS. In addition, by producing pattern dependency graphs, our mechanism is able to assign obfuscation techniques to conceal sensitive patterns selectively and make an obfuscation plan ahead of time. Our evaluation results demonstrated that APP-CEP outperforms two baseline techniques, namely Window-based and Graph-based approaches, in the total percentage of revealed private pattern matches and performed slightly better in the number of detection errors of public pattern matches. Moreover, by involving more background knowledge, APP-CEP achieves a significant performance in the expected revealed private matches by the adversary. Furthermore, the PUT results prove that APP-CEP successfully provides a solution to the gap mentioned in this study and envisions a new era in this field for adaptively assigning obfuscation techniques regardless of actual events in the streams effectively.

In our future work, we will consider designing an intelligent algorithm to select the best obfuscation technique in case the dependency sub-graphs do not offer a ZOD node. That can be achievable by introducing a comparison metric by which both positive and negative effects of obfuscations can be computed and labeled in the sub-graphs. Besides, Enhancing the suitability for real-world applications requires thoughtful consideration of the stakeholders and their respective interests when selecting input representations. Moreover, we plan to extend our evaluation to different use cases to figure out other aspects of this problem as well as APP-CEP limitations in fulfilling various applications' requirements.

## ACKNOWLEDGEMENTS

This work has been supported by the Research Council of Norway as part of the project Parrot (311197).

## REFERENCES

Alt, B., Weckesser, M., Becker, C., Hollick, M., Kar, S., Klein, A., Klose, R., Kluge, R., Koeppl, H., Koldehofe, B., et al. (2019). Transitions: A protocol-independent view of the future internet. *In Proceedings of the IEEE*, 107(4):835–846.

Barber, F. and Furnell, S. (2022). Benchmarking consumer data and privacy knowledge in connected and autonomous vehicles. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP'22)*, pages 426–434.

Chen, D., Sain, S. L., and Guo, K. (2012). Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19:197–208.

Flink (2017). Flink-cep. https://nightlies.apache.org/flink/flink-docs-master/docs/libs/cep/, Accessed on Oct 9th, 2023.

Gundersen, M. (2022). Statistics norway demands to know exactly what norwegians buy in the grocery store. https://nrkbeta.no/2022/05/28/ssb-krever-a-fa-vite-noyaktig-hva-nordmenn-kjoper-i-matbutikken/, Accessed on Oct 9th, 2023.

He, Y., Barman, S., Wang, D., and Naughton, J. F. (2011). On the complexity of privacy-preserving complex event processing. In *Proceedings of the 13th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 165–174.

Leicht, J. and Heisel, M. (2023). P2bac: Privacy policy based access control using p-lpl. In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP'23)*, pages 686–697.

Li, L., Zhong, B., Hutmacher Jr, C., Liang, Y., Horrey, W. J., and Xu, X. (2020). Detection of driver manual distraction via image-based hand and ear recognition. *Accident Analysis & Prevention*, 137:p. 105432.

Lotfian Delouee, M., Koldehofe, B., and Degeler, V. (2022). Towards adaptive quality-aware complex event processing in the internet of things. In *Proceedings of the 18th International Conference on Mobility, Sensing and Networking (MSN'22)*, pages 571–575. IEEE.

Lotfian Delouee, M., Koldehofe, B., and Degeler, V. (2023a). Aqua-cep: Adaptive quality-aware complex event processing in the internet of things. In *Proceedings of the 17th ACM International Conference on Distributed and Event-Based Systems (DEBS'23)*, page 13–24. ACM.

Lotfian Delouee, M., Koldehofe, B., and Degeler, V. (2023b). Poster: Towards pattern-level privacy protection in distributed complex event processing. In *Proceedings of the 17th ACM International Conference on Distributed and Event-Based Systems*, page 185–186. ACM.

Palanisamy, S. M. (2020). Towards multiple pattern type privacy protection in complex event processing through event obfuscation strategies. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2020 International Workshops, DPM 2020 and CBT 2020, Guildford, UK, September 17–18, 2020, Revised Selected Papers 15*, pages 178–194. Springer.

Palanisamy, S. M., Dürr, F., Tariq, M. A., and Rothermel, K. (2018). 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*, pages 40–51.

Plagemann, T., Goebel, V., Hollick, M., and Koldehofe, B. (2022). Towards privacy engineering for real-time analytics in the human-centered internet of things. *arXiv preprint arXiv:2210.16352*.

Schilling, B., Koldehofe, B., Rothermel, K., and Ramachandran, U. (2013). Access policy consolidation for event processing systems. In *2013 Conference on Networked Systems*, pages 92–101. IEEE.

Stach, C. and Steimle, F. (2019). Recommender-based privacy requirements elicitation-epicurean: an approach to simplify privacy settings in iot applications with respect to the gdpr. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1500–1507.

Tokas, S., Erdogan, G., and Stølen, K. (2023). Privacy-aware iot: State-of-the-art and challenges. In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP'23)*, pages 450–461.

Tran, T., Nguyen, P., and Erdogan, G. (2023). A systematic review of secure iot data sharing. In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP'23)*, pages 95–105.

Wang, D., He, Y., Rundensteiner, E., and Naughton, J. F. (2013). Utility-maximizing event stream suppression. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 589–600.