Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems

Mikhail Fomichev*
TU Darmstadt
mfomichev@seemoo.tu-darmstadt.de

Manisha Luthra*
TU Darmstadt & DFKI
manisha.luthra@dfki.de

Maik Benndorf
University of Oslo
maikb@ifi.uio.no

Pratyush Agnihotri
TU Darmstadt
pratyush.agnihotri@kom.tu-darmstadt.de

ABSTRACT

Stream processing systems (SPSs) have been designed to process data streams in real-time, allowing organizations to analyze and act upon data on-the-fly, as it is generated. However, handling sensitive or personal data in these multilayered SPSs that distribute resources across sensor, fog, and cloud layers raises privacy concerns, as the data may be subject to unauthorized access and attacks that can violate user privacy, hence facing regulations such as the GDPR across the SPS layers. To address these issues, different privacy-preserving mechanisms (PPMs) are proposed to protect user privacy in SPSs. Yet, selecting and applying such PPMs in SPSs is challenging, since they must operate in real-time while tolerating little overhead. The multilayered nature of SPSs complicates privacy protection because each layer may confront different privacy threats, which must be addressed by specific PPMs. To overcome these challenges, we present Prinseps, our comprehensive privacy vision for SPSs. Towards this vision, we (1) identify critical privacy threats on different layers of the multilayered SPS, (2) evaluate the effectiveness of existing PPMs in addressing such threats, and (3) integrate privacy considerations into the decision-making processes of SPSs.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Information systems** → *Stream management*.

KEYWORDS

Stream processing, Privacy, Threat modeling, Access control, IoT

1 INTRODUCTION

Motivation. *Can we create a world without a disease?* This question was posed by Max Welling—a renowned machine learning (ML) scientist in his visionary talk.[1] He underlined two requirements to achieve this goal: (1) privacy-preserving systems analyzing human-centric sensor data and (2) foundations in deep learning that enable better diagnosis and treatment of diseases. This work makes a step to fulfill the first requirement towards this noble goal using stream processing systems (SPSs). SPSs can serve as a core technology for analyzing sensor data collected by Internet of Things (IoT) devices in real-time. Recently, SPSs have adopted a *multilayered approach* by distributing processing resources between *sensor*, *fog*, and *cloud* layers to achieve lower processing latency and better utilization of resources [29]. Hence, we focus on multilayered SPSs in this work.

The data streams that originate by sensing users or their environment contain inherently sensitive information, e.g., user's lifestyle, requiring SPSs that process such data to comply with privacy regulations, like the GDPR [25]. As crucially, users start to increasingly demand stricter control over the data collected by IoT applications about them [13]. These points make privacy protection critical for the success of SPSs in the future.

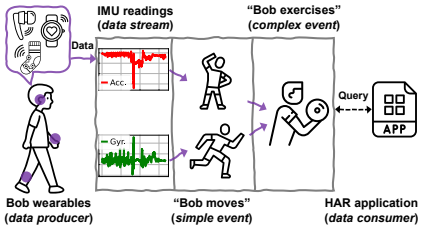

Figure 1: Typical workflow of an SPS based on our running example of a human activity recognition (HAR) application enabled by wearable sensors.

Running Example. To concertize our vision for privacy in multilayered SPSs, we pick a human activity recognition (HAR) application enabled by wearable devices as our running example in this work (cf. Figure 1). Specifically, a user, Bob, carries wearable devices (e.g., a smartwatch) known as *data producers*, which collect inertial measurement unit (IMU) data, i.e., accelerometer and gyroscope readings. Utilizing these data, the HAR application, called the *data consumer*, can monitor Bob's activity, like exercising, for medical and health insurance purposes. This happens as follows: the stream of IMU data is input to an SPS, which first classifies these data into *simple events* (e.g., "Bob moves") and then combines such simple events to detect *complex events*, like "Bob exercises". The HAR application is mainly interested in monitoring these complex events, hence it defines a *continuous query* to identify them in the ceaseless stream of the IMU data.

Our running example exhibits that the SPS scrutinizes Bob's data on different granularity: from raw IMU data to complex events. This clearly has privacy concerns, like the HAR application can detect not only *public* complex events (or patterns) that are needed for its functionality, e.g., "Bob exercises", but also *private* complex events (or *patterns*), like "taking medicine" by Bob, which is revealed as a sequence of "swallow" → "drink" → "lay down" events.

Contributions. The topic of privacy protection in SPSs has recently emerged, yet the privacy issues stemming from SPS's multilayeredness remain underexplored [19, 25]. As such, the privacy-preserving mechanisms (PPMs) utilized in SPSs are typically adopted from the database community, requiring a major redesign to satisfy the real-time nature of SPSs [18], while several PPMs developed specifically for SPSs are still not mature, which makes it difficult to holistically apply them in SPSs [25]. We identify *three challenges* of using existing PPMs in multilayered SPSs: (1) threats to user privacy are not systematically understood in these SPSs, complicating the selection of appropriate PPMs; (2) PPMs must *not only* be applied holistically,

*Both authors contributed equally to this work.
[1]TEDx talk: https://youtu.be/g9HA8A8tEUs [Accessed on 26.05.2023].

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

i.e., at each layer of the multilayered SPS to provide strong privacy protection, *but also* satisfy real-time needs of SPSs, imposing little overhead; (3) the PPMs, selected and applied in this fashion, should enable a *privacy-utility tradeoff (PUT)*, which is tunable depending on SPS's user privacy needs and application *quality of service (QoS)* requirements. These challenges unfold *two research questions* that we seek to answer in this work:

RQ1: What are the most critical threats to user privacy in multilayered SPSs used in IoT applications?

RQ2: How to systematically select, customize, and apply PPMs that address these threats while enabling a PUT?

Addressing the above questions and inspired by [19], we present our vision for holistic privacy protection in multilayered SPSs called Prinseps—*Privacy in Stream Processing Systems*. To pursue this vision, we (1) review the state of privacy protection in SPSs (Section 2); (2) identify critical privacy threats and their conduct in multilayered SPSs, which allows us to shape Prinseps' design (Section 3); and (3) provide the preliminary evaluation of Prinseps along with further research opportunities (Section 4). Despite focusing on the IoT use cases for SPSs to concretize our vision, we perform the above three steps in a generic manner, making Prinseps generalizable to other SPSs' applications. Our results show that there is *no* one-size-fits-all solution for privacy in SPSs, i.e., different PPMs have their up- and downsides (which we investigate), while being applicable in various parts of the multilayered SPS.

2 RELATED WORK

Privacy in Stream Processing. The subject of privacy protection in SPSs has started receiving attention from the research community, yet existing works mainly focus on preserving privacy in *one* of the following directions in the SPS: (1) raw data streams or (2) private patterns, whereas a holistic view on privacy is still missing. For *raw data streams*, prior research *adopted* PPMs from the database community, like differential privacy (DP), k-anonymity, and l-diversity [4, 5, 10]. Such PPMs seek to protect single events (e.g., "Bob moves" from Figure 1) by grouping them into clusters—within which these events become indistinguishable for the data consumer of an SPS, like a HAR application. There exist two issues with such adopted PPMs: (1) they introduce extra overhead due to clustering, hindering the real-time performance of SPSs [8]; (2) most works assume that sensitive events are revealed via simple *count* queries [4, 5, 20], which does not reflect real-world SPSs—where both more complex *queries on* raw data streams and *access to* such streams can become
attack vectors to violate user privacy [6, 14].

There exist a few PPMs protecting *private patterns*, e.g., "taking medicine" by Bob. [18] presents a PPM leveraging event reordering to conceal private patterns. In [6], a PPM that enforces access control (AC) on private patterns is proposed. This PPM estimates trust among nodes within an SPS, allowing only trusted nodes to access private patterns. While being important preliminary research, the PPMs [18] and [6] lack robust privacy guarantees and rely on strong assumptions, like the availability of trust information in an SPS.

A related line of work to privacy in raw data streams and private patterns is *privacy-preserving stream analytics*, which enforces AC on SPS's queries by means of cryptography [3]. While these PPMs protect against malicious attackers, they become less relevant under our threat model (cf. Section 3), where such PPMs using encryption and authentication are assumed to be in place. Similar to the stream analytics, *privacy preservation in publish/subscribe systems* heavily relies on AC-based PPMs to shield sensitive data flows [17]. Another trend in privacy protection that benefits all of the described PPMs is leveraging trusted execution environments (TEEs) to safeguard the runtime of PPMs within an SPS against malicious components [24].

Privacy-utility Tradeoff. The above PPMs only loosely consider a PUT when applied in SPSs, i.e., the impact of a PPM is, in the *best case*, evaluated as an interplay between one privacy and one utility metric in a specific scenario. Yet, the tradeoff between these metrics, their fine-tuning, and generalizability to other scenarios receive little attention. A few recent works present first systematic approaches to balance a PUT, which can either be achieved via an optimization problem [1] or through user input [7]. However, the applicability of such approaches has *not yet* been explored in the context of SPSs.

Research Gap. As shown above, the privacy protection in SPSs has been applied in an *ad-hoc fashion*, adopting existing or devising new PPMs to address isolated privacy issues. This happens due to a limited understanding of critical privacy threats and their occurrence inside SPSs. Thus, it is difficult to justify the selection, customization, and application of PPMs addressing those threats while preserving the utility of SPSs. Without such justification, it remains unclear *where and how* to apply even the golden standard PPMs, like DP [21].

3 PRINSEPS: VISION FOR PRIVACY IN SPS

To approach the vision of Prinseps, we take four steps: (1) identify critical privacy threats and their conduct in a multilayered SPS; (2) use this knowledge to select exemplary PPMs that we evaluate to find their suitability for Prinseps; (3) show how to customize these PPMs, leveraging user privacy policies to facilitate a PUT; and (4) formulate the principles of applying PPMs in Prinseps to enable a holistic privacy protection in multilayered SPSs.

We envision the following *success criteria* for Prinseps: (1) simultaneous protection against critical privacy threats in a multilayered SPS, allowing its users to control the protection granularity as well as fine-tune the PUT; (2) seamless extendibility of Prinseps with new PPMs to address existing and/or novel privacy threats.

Privacy Threats. To identify critical privacy threats and how they occur within a multilayered SPS, whose architecture is shown in Figure 2, we use our running example. In Figure 2, the IMU *data flows* from Bob's wearables, e.g., a smartwatch (*sensor layer*), up into fog nodes, like a smart hub, where these data are classified into simple events, e.g., "Bob moves" (*fog layer*), that are pushed up to the cloud, where complex events, like "Bob exercises", are ultimately inferred from such aggregated simple events (*cloud layer*). To monitor Bob's activity, the HAR application queries the SPS—we assume the *query flow* can happen between the HAR application and *any* of the SPS's layers: sensor, fog, and cloud (cf. dotted arrow lines in Figure 2); this gives flexibility to SPSs, e.g., in terms of query result granularity.

We use an *honest-but-curious (HBC)* adversary model to identify critical privacy threats in multilayered SPSs, due to its realism: here, we assume that an SPS node on each layer (sensor, fog, cloud) and data consumer (HAR application) can be HBC [18, 20]. Such nodes and data consumers conform to SPS's functionality, i.e., they detect

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

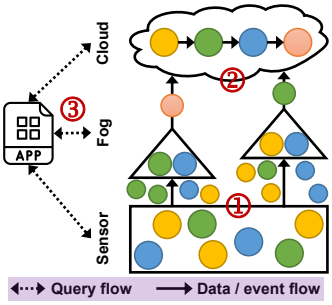

Figure 2: Multilayered SPS architecture used in Prinseps, consisting of nodes placed on sensor (▭), fog (△), and cloud (☁) layers; ●, ●, ● denote data or simple events from different sensors, while ● is an inferred complex event. ① to ③ mark the occurrence of critical privacy threats in this architecture.

expected events faithfully and do not violate the correctness of queries but seek to learn extra information about Bob, like habits, invading his privacy. We differentiate *attack vectors* between HBC nodes and data consumers: the former scrutinize Bob's sensor data and events during their processing, while the latter craft queries to reveal Bob's sensitive events, e.g., "taking medicine".

Based on our threat model, we conduct research utilizing related work analysis, self-expertise, discussions with privacy practitioners, and review of best privacy practices[2] to determine generic privacy threats in multilayered SPSs: (1) *sensitive attributes*, (2) *private patterns*, and (3) *invasive queries*. Please note that the found three threats are not claimed to be exhaustive, and we presume the PPMs preventing malicious attackers, like encryption (cf. Section 2), to be deployed—such PPMs alone cannot thwart HBC adversaries which could, e.g., legitimately access encryption keys. In the following, we detail the three privacy threats identified in multilayered SPSs, as they help us to shape the design of Prinseps.

(1) Sensitive Attributes. The sensitive attributes, like Bob's weight, can be inferred from a few seconds of IMU data using deep learning [11]. Such amount of data is available on sensors and fog devices (e.g., smartwatch and smart hub), whose hardware is performant to carry inferences of sensitive attributes in real-time [11]. Hence, this threat happens on HBC sensor- and fog-layer nodes while processing of Bob's IMU data (cf. ① in Figure 2). When choosing the PPMs to tackle the inference of sensitive attributes, we try to achieve *low latency* as our primary utility metric of an SPS since low latency is the main advantage of processing data near its producers [29].

(2) Private Patterns. The private patterns, such as "taking medicine" by Bob, are complex events detectable by an SPS via sequence matching of (simple) events comprising this pattern. Hence, this threat materializes in fog and cloud layers, where the chronological sequences of events are available (cf. ② in Figure 2). We concur with prior research that issuing HBC queries to fog and cloud nodes is the most feasible strategy for revealing private patterns [18]. When selecting the PPMs to conceal private patterns, we seek *high accuracy* of public complex events (e.g., "Bob exercises") that should be detectable by an SPS as our main utility metric.

(3) Invasive Queries. The invasive queries aim to reveal sensitive information about Bob, like his lifestyle, going beyond private patterns. Such threats happen during query processing by HBC nodes which can, e.g., augment a query on Bob's exercising with the simple filter condition: `time > 9am AND time < 6pm` to disclose the fact of Bob's exercising during working hours to his employer. Because Prinseps allows queries to be processed by the sensor, fog, and cloud layers, this threat can occur on any of them (cf. ③ in Figure 2). When choosing the PPMs to address invasive queries, we consider *both* low latency and high accuracy of public (complex) events as our target utility metrics of an SPS.

[2]These steps follow the current state of the art on privacy threat modeling [2].

Table 1: Requirements for formulating privacy policies in Prinseps.

Requirement	Explanation
R1: Transparency	Users understand purposes for which their data are utilized.
R2: Accessibility	Users configure their privacy in an easy and intuitive manner.
R3: Proactiveness	Users are assisted in configuring their privacy.
R4: Awareness	Users are made aware of privacy threats they did not consider.
R5: Granularity	Users control their privacy in a fine-grained manner.

Exemplary PPMs for Prinseps. After describing critical privacy threats and their conduct in multilayered SPSs, we utilize (1) assets that need protection, (2) target utility metrics, and (3) the computing capabilities of SPS's layers as our criteria to select exemplary PPMs for Prinseps. We identify *three types of PPMs* that satisfy our criteria while being frequently used by real-world applications [7, 11]: machine learning (ML)-, differential privacy (DP)-, and access control (AC)-based PPMs. We showcase how these PPMs address sensitive attributes, private patterns, and invasive queries threats in multilayered SPSs (cf. Section 4), shedding light on which other information can be utilized by Prinseps to select PPMs.

Privacy Policies: Customizing PPMs to Enable the PUT. There exists clear evidence that PPMs can improve their PUT by protecting the privacy of "what users care for" [7, 27]. Hence, identifying *user privacy needs* can help Prinseps to customize PPMs, making their parameterization and application *selective*. This enhances utility, as protection levels of PPMs (e.g., amount of noise added to data) and their usage frequency are limited to user privacy needs, which also allows fine-tuning the PUT.

Prinseps captures users' privacy needs in the form of a privacy policy. The literature shows various requirements for a user-defined privacy policy [13, 25]—we extract the most relevant for Prinseps in Table 1. We create the privacy policy based on *static* and *dynamic rules* [25], as explained using our running example. Respecting **R1:** *Transparency*, a HAR application must brief Bob that IMU data from his wearables is collected to monitor his exercise. Bob can also be prompted to provide his weight for a better QoS—if he decides *not* to disclose it, then such a privacy need must be intuitively expressive, satisfying **R2:** *Accessibility*.

Prinseps utilizes *policy rules* to map Bob's privacy needs, like a static policy rule to conceal his weight. Such policy rules can be predefined in Prinseps by domain privacy experts, e.g., in the form of trigger-action statements. Prinseps finalizes Bob's privacy policy by clarifying and making suggestions to select the best-matching policy rules, as per **R3:** *Proactiveness*. Despite hiding his weight, Bob must be alarmed that *its inference* is still feasible via IMU data, fulfilling **R4:** *Awareness*. Hence, Prinseps offers Bob to either apply a PPM(-s) against this threat or accept it. In the first case, Bob gets informed about PPM's impact on utility, and he is guided towards tuning his PUT, following **R5:** *Granularity*. For example: the used PPM can, by default, reduce weight inference to 10% at the cost of recognizing 90% of Bob's activity correctly; Bob can adjust this

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

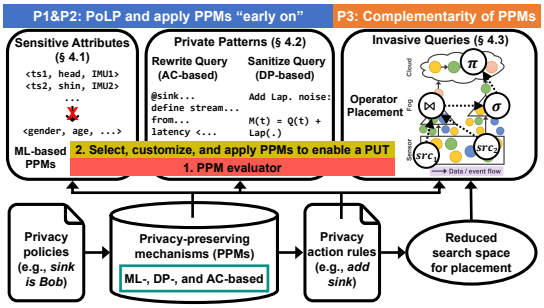

Figure 3: Overview of Prinseps privacy-preserving architecture for SPSs.

PUT to, e.g., 1% weight inference vs. 80% of activity recognition accuracy.

Bob may need to change the usage of a specific PPM, like its PUT, based on his *context*, e.g., time and location. Prinseps achieves this via dynamic rules that override Bob's static policy rules, relying on his feedback in a situation, e.g., Bob allows the HAR application to access his weight *only if* he is at a doctor's office.

Principles of Applying PPMs. To enable holistic privacy protection in SPSs, the application of PPMs in a *privacy-by-design* manner is as vital as PPMs' privacy guarantees [25]. Prinseps follows three such privacy-by-design principles: **P1:** the *Principle of least privilege (PoLP)*, **P2:** *Applying a PPM "early on"*, and **P3:** *Complementarity of PPMs*. With **P1**, data minimization is ensured, like prohibiting smart home apps from accessing raw sensor data, since they only require high-level events for their functionality [13]. Using **P2**, we demand that a PPM is applied *as close* to the source of the threat *as possible*, thus, it will not propagate (e.g., to the next layer in an SPS) or escalate. **P3** stipulates that each known threat must, by default, be treated by a PPM. Yet, complex threats may not be preventable by a single PPM: in this case, the combination of PPMs that reinforce each other should be applied to tackle the complex threats. To *avoid circumventing* PPMs used in Prinseps via misconfiguration, we, on top of **P1**–**P3**, demand such PPMs to be *deployed as per best security practices*, like running PPMs inside TEEs and obfuscating them [2].

4 SHAPING PRINSEPS: RESULTS

Figure 3 shows an overview of Prinseps whose goal is to *holistically* address critical privacy threats in multilayered SPSs while enabling a PUT. Prinseps has the following workflow: the *pool of (exemplary) PPMs* tackling these critical threats is maintained, i.e., novel PPMs are included as new privacy threats appear, and outdated PPMs get removed. From this pool, the Prinseps' component called the *PPM evaluator* picks a candidate PPM to address a specific privacy threat based on *criteria* such as: privacy guarantees, runtime performance, utility metrics, resource requirements, scalability, and ease of setup; this list is non-exhaustive. Then, Prinseps customizes the candidate PPM leveraging a *user privacy policy* to fulfill his/her privacy needs while minimizing the impact on utility by restricting PPM's usage to such privacy needs and its level of protection—to the user's preferences. Finally, the customized PPM is deployed at a specific SPS layer, i.e., sensor, fog, or cloud (cf. Figure 2).

The workflow of Prinseps is *dynamic*, i.e., each component can receive updates and be rerun, like including a new PPM to the pool can trigger the evaluator to reassess this PPM as a better candidate to tackle a privacy threat or adding a fresh privacy policy rule will require Prinseps to re-customize the PPM(-s) already in use.

In the following, we report our findings on how Prinseps can utilize state-of-the-art ML-, DP-, and AC-based PPMs to address sensitive attributes, private patterns, and invasive queries threats (cf. Section 3). We also discuss open challenges and research opportunities to shape Prinseps and the privacy landscape in SPSs more broadly. Sections 4.1 through 4.3 cover each of the above privacy threats and are *structured as follows*: we (1) justify the selection of the most suitable PPM(-s) against the threat, using our principles of applying PPMs listed in Section 3, findings from related work, and self-expertise; (2) report evaluation results for each PPM tackling the corresponding privacy threat; and (3) present opportunities for future research.

4.1 Conceal Sensitive Attributes

PPM Selection. Among ML-, DP-, and AC-based PPMs, we see that the former fits best to conceal sensitive attributes, as they: (1) work directly on raw data streams (**P2**), minimizing the information on the sensitive attribute, like weight (**P1**); and (2) show near real-time performance [11], allowing Prinseps to meet the utility goal related to this threat—low latency. Contrarily, DP-based PPMs target raw data streams by adding noise to query results on such data (cf. Section 2), violating **P2**, whereas injecting noise directly into raw data either degrades its accuracy (and thus utility) or leads to significant noise scalability issues [8]. The AC-based PPMs could, in principle, conceal sensitive attributes by limiting access to parts of raw data, satisfying **P1** and **P2**. Yet, sensitive attributes are embedded in raw data (e.g., gender in the IMU data of Bob) [11], making it difficult to formulate such AC-rules.

To see how Prinseps can utilize ML-based PPMs, we review one recent PPM—*ObscureNet*, that uses the *variational autoencoder (VAE)* neural network to transform IMU data, hiding sensitive attributes in near real-time [11]. This PPM reduces the success of inferring a sensitive attribute, like gender, from 90% to 50% while keeping the accuracy of public HAR events searched by an SPS above 90%.

Results: ML-based PPMs. We seek to understand the (1) PUT and (2) runtime performance of such PPMs in the real-world SPS case of several wearable devices, as per our running example. Therefore, we evaluate ObscureNet on the *RealWorld (HAR)* dataset that contains IMU data from *five devices* located as such: head, upper arm, waist, thigh, and shin [26]; originally, ObscureNet is validated on the IMU data coming from a single thigh-worn device [11].

For comparability, we use ObscureNet[3] with the same parameters as in [11]; the sampling rate of IMU data from RealWorld (HAR) is 50 Hz. Table 2 shows the *gender inference* accuracy on the five body locations (i.e., head to shin) for four HAR events: walking, standing, jogging, and climbing upstairs, which must be detected by the SPS. We see that gender inference drops from over 90% to below 20% after applying ObscureNet, being in line with [11]. This behavior is consistent across different body locations and HAR events; our results for *weight inference* are similar. Nevertheless, we reveal that ObscureNet can lower the accuracy of HAR events by up to 50% (cf. Table 3), harming the SPS's utility, which was unseen in [11].

[3]https://github.com/sustainable-computing/ObscureNet [Accessed on 26.05.2023].

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

Table 2: Gender inference accuracy on five body locations for four HAR events before and after applying ObscureNet.

Accuracy / Event / Location	Walking (b / a, %)	Standing (b / a, %)	Jogging (b / a, %)	Upstairs (b / a, %)
Head	99.1 / 14.7	92.3 / 16.1	98.9 / 17.3	97.9 / 17.4
Upper arm	99.5 / 7.1	94.0 / 41.0	98.3 / **10.8**	99.0 / 22.5
Waist	99.4 / 13.1	90.2 / **15.3**	93.8 / 20.6	98.1 / 35.3
Thigh	95.0 / **6.2**	80.0 / 31.2	91.3 / 12.0	95.5 / **16.8**
Shin	92.1 / 20.1	73.0 / 45.2	93.1 / 28.7	92.6 / 22.1
Combined	97.0 / 29.4	85.9 / 33.6	95.2 / 36.7	94.7 / 36.9

(b / a, %) – inference accuracy in % *before* and *after* applying ObscureNet.

Table 3: Event inference accuracy on five body locations for four HAR events before and after applying ObscureNet.

Accuracy / Event / Location	Walking (b / a, %)	Standing (b / a, %)	Jogging (b / a, %)	Upstairs (b / a, %)
Head	91.1 / 39.0	90.0 / 66.4	98.4 / 83.4	98.0 / 89.5
Upper arm	92.4 / 66.8	90.8 / 78.4	95.6 / 87.3	96.4 / 86.1
Waist	97.7 / **93.7**	82.7 / 69.8	98.4 / 89.6	97.8 / 80.0
Thigh	92.0 / 75.8	92.9 / 66.1	95.5 / 88.7	97.6 / 95.1
Shin	92.6 / 60.5	94.3 / **92.8**	94.7 / 87.9	96.6 / **94.4**
Combined	90.5 / 37.3	82.1 / 62.4	93.7 / **87.6**	93.5 / 83.5

(b / a, %) – inference accuracy in % *before* and *after* applying ObscureNet.

We assess the runtime of ObscureNet for several devices reusing the benchmark of [11] carried on off-the-shelf hardware: Raspberry Pi 3. From this benchmark, we calculate the time budget for running ObscureNet on our IMU data to be 200 ms, while ObscureNet needs around 100 ms to conceal sensitive attributes on these data. Hence, for the IMU data of 50 Hz, ObscureNet can protect the privacy of data streams from *two devices* simultaneously in near real-time.

Research Opportunity. Our results indicate that ML-based PPMs, like ObscureNet, favor privacy over utility in the realistic SPS case with several wearables. Hence, utility—*both* in terms of accuracy of HAR events and runtime performance (i.e., for lower latency)—needs to be improved, which requires an *in-depth study* of such recent PPMs to adjust their PUT. This twofold utility can be enhanced through *federated learning*, leveraging data streams from different devices [28]. The runtime performance of ObscureNet boosts by lowering the data sampling rate, like by two times if we go from 50 to 25 Hz. Thus, ML-based PPMs should be further explored towards utilizing task-aware knowledge. Hereby, we spot another research avenue—studying Prinseps' *generalizability*—to investigate sensitive attributes in non-human-centric environments (e.g., smart factory) and evaluate the efficacy of existing ML-based PPMs in concealing such sensitive attributes.

4.2 Protect Private Patterns

PPM Selection. From ML-, DP-, and AC-based PPMs, we find that the latter two are best suited to protect private patterns, like "taking medicine" in our running example. The reasons are that both PPMs have been used on queries [12, 27] while satisfying **P1** and **P2**. Considering DP-based PPMs, they minimize private information in the query result by *adding noise* to it (sanitization) *after* processing the query but before returning its response [27]. While AC-based PPMs *rewrite queries before* processing them, restricting access to sensitive data leaked by the query result [12]. Thus, these PPMs can also be complementary, as per **P3**. We have not found any ML-based PPMs that are applicable on queries to protect private patterns.

Table 4: Instantiating Swellfish privacy to conceal private patterns in Prinseps.

Event / Time / Day	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Bob's Day 1	swallow	drink	lay d.	drink	swallow	lay d.	walk
Bob's Day 2	walk	swallow	drink	lay d.	walk	lay d.	drink
...	...	...	...	...	...	...	...
Bob's Day n	swallow	drink	lay d.	walk	swallow	drink	lay d.
$Q(t)$	–	–	2	1	0	0	1
$Lap(\cdot)$	$\frac{n\cdot\epsilon}{3}$	$\frac{n\cdot\epsilon}{3}$	$\frac{n\cdot\epsilon}{3}$	$\frac{n\cdot\epsilon}{3}$	$\frac{n\cdot\epsilon}{2}$	$n\cdot\epsilon$	0

– relevance interval; – private pattern; small $Lap(\cdot)$ means more noise, n – num. of days.

Results: DP-based PPMs. We (1) determine the most suitable class of DP-based PPMs for protecting private patterns and (2) instantiate one such PPM [27], exhibiting how it can be leveraged by Prinseps.

To date, there exist three classes of DP-based PPMs adding noise at the granularity of: events, windows of events, and users—denoted as *event-level*, *w-event level*, and *user-level* PPMs [21]. The first class targets single events, e.g., "Bob drinks", and not their relationships, making these PPMs unsuitable for our purpose. As for the user-level DP, it protects the entire data stream of a person but requires many users to be involved, rendering such PPMs infeasible in single-user cases (as our running example) due to a reduced PUT [21]. Instead, we study w-event level PPMs that conceal a series of events within a time window, allowing a better PUT for protecting private patterns.

We instantiate a novel w-event level PPM: *Swellfish privacy* [27], for our running example to assess its impact on the PUT. This PPM improves on existing w-event level DP methods by adding noise to queries *selectively*, i.e., only within a *relevance interval* provided by a user (e.g., Bob), and *not* universally to every window of length w events occurring in the data stream. The relevance interval specifies a timeframe inside which a private pattern can happen, like "taking medicine" by Bob around mealtimes. Being selective in adding noise, allows [27] to attain between *one to three orders of magnitude* better PUT than the state-of-the-art on w-event level DP. Swellfish privacy is thus an exemplary PPM, showcasing how utilizing a user privacy policy can facilitate the PUT, as advocated by Prinseps.

We determine that Swellfish privacy notably benefits the PUT in Prinseps being instantiated to protect the private pattern of "taking medicine" by Bob over multiple days. Thus, a HAR application that is curious about Bob's health condition cannot discern if Bob takes medicine on a specific day. As in [27], the private pattern is revealed via *count queries* over predefined Bob's events, like $[walk, swallow, drink, lay\ down]$, where each query returns the occurrences of such events, e.g., $[1, 0, 0, 0]$, performed by Bob at a point in time.

Table 4 shows how Prinseps hides the private pattern of "taking medicine" utilizing the adapted DP-based PPM of Swellfish privacy. The main idea is to inject Laplacian noise into the query result $Q(t)$ *only within* the relevance interval inside which the "taking medicine" pattern occurs (as a sequence of "swallow" → "drink" → "lay down" events) to protect it while adding no noise to the result outside the relevance interval to preserve utility. This allows us to prevent the detection of at least one simple event, like "drink", hence concealing the private pattern. Specifically, $M(t) = Q(t) + Lap(\cdot)$ is the *sanitized query result* utilizing this PPM, where $Lap(\cdot)$ shows the amount of added Laplacian noise. We set the size of the relevance interval to *four timestamps* to encompass the length of our private pattern (i.e., $w = 3$ in the w-event level DP terminology). Other

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

parameters that include *global query sensitivity* and *privacy budget* ϵ are chosen to be one and the range 0.1–10, respectively [27].

The noise reduction from the privacy budget of $\frac{\epsilon}{3}$, equally distributed in the relevance interval $[t_1, t_4]$, to 0 outside this relevance interval at t_7 in $[t_5, t_7]$ improves utility (i.e., the detection accuracy of public complex events, like "Bob exercises") without comprising privacy, leading to a better PUT. However, the rapid noise reduction outside the relevance interval is not well-studied even in [27]—not to mention the PUT behavior in various other SPS use cases.

Results: AC-based PPMs. We exemplify how Prinseps can achieve AC on private patterns. For this, it relies on a *query rewriting* component that takes a query submitted by a data consumer (e.g., HAR application) and enforces *privacy action rules* on the query by rewriting it if necessary. These privacy action rules define *how* the query is rewritten, and they are instantiated from the user privacy policy with respect to the PPM (cf. Figure 3). Note that the query rewriting component is the *entry point* of the query in Prinseps, ensuring **P2** and following best practices for query rewriting in databases [14].

We show how Prinseps enforces AC using our *running example count query*, where "swallow" → "drink" → "lay down" events must occur, e.g., within a window of two minutes, to reveal the private pattern of "taking medicine" by Bob:[4]

```
define stream TakeMedicineStr (ts long, cnt_swallow int,
cnt_drink int, cnt_layd int);
from every e1=TakeMedicineStr[ user_activity == 'swallow' ]
     -> e2=TakeMedicineStr[ user_activity == 'drink' ]
     -> e3=TakeMedicineStr[ user_activity == 'lay␣down' ]
    within 2 min
select e3.ts, count(e1.user_activity) as cnt_swallow,
count(e2.user_activity) as cnt_drink,
count(e3.user_activity) as cnt_layd
insert into TakeMedicinePattern;
```

Upon seeing such a query—issued by the HAR application—the query rewriting component of Prinseps first checks if there exist privacy action rules related to this query. Imagine Bob has defined a privacy policy rule demanding to forward the results of queries, searching for the "taking medicine" pattern, *only* to himself. In this case, the HAR application sends the query to detect these patterns, violating Bob's privacy policy. This violation is noticed by the query rewriting component, triggering the action rule to rewrite the `@sink` clause such that the query result is only presented to Bob:

```
@sink(publisher='Bob')
```

With this, Bob's private patterns become protected while preserving utility, i.e., the detection accuracy of his public complex events, like "Bob exercises", improving the PUT.

Research Opportunity. We identify a few challenges to the protection of private patterns in Prinseps. First, both DP- and AC-based PPMs need to be benchmarked on the set of private patterns to understand PPMs' *efficiency and tradeoffs*, e.g., in terms of protection levels and runtime performance. Second, we see the potential for these two PPMs to *complement each other* to protect private patterns, materializing our **P3**. While we are not aware of any exemplary architectures for combining PPMs in SPSs, there exist some in databases [14]. Third, we view the *generalizability* of private patterns' protection in Prinseps, as another research avenue, like discovering and preserving private patterns in enterprise environments, where such patterns can leak intellectual property. This research opportunity is especially topical, given a revived interest in the protection of private patterns [9].

[4]We use Siddhi query language syntax here: https://siddhi.io [Accessed on 03.03.2023].

4.3 Mitigate Invasive Queries

PPM Selection. We perceive invasive queries as a complex privacy threat, hence it is unlikely to be addressed by any of the PPMs used by Prinseps (i.e., ML-, DP-, and AC-based) in their traditional form alone. Combining different PPMs as per **P3** (e.g., DP and AC) should help to tackle such complex threats, but it remains a difficult task (cf. Section 4.2), with little research done in that direction [14]. Hence, we explore how the *operator placement* in SPSs, defining *where* and *how* the query operators are processed within the multilayered SPS [15], can be leveraged as the AC-based PPM against HBC SPS nodes (cf. Section 3), mitigating invasive queries close towards the source of the threat, satisfying **P2** and implicitly **P1**.

Results: AC-based PPMs. We sketch how operator placement can be utilized in Prinseps to address invasive queries. To date, the operator placement algorithms focus only on *utility-related metrics*, like latency and throughput, to decide which SPS nodes will process the queries [15, 16]. There exists limited work that considers *privacy* to inform the operator placement, e.g., [23] combines privacy and utility goals in the placement decisions, but only from the viewpoint of designing a query language for SPSs, while [6] incorporates user's trust into an SPS node to the cost model, however assessing trust of resources in a multilayered SPS, whose properties are hidden from end users, is infeasible. In contrast, Prinseps approaches the task of operator placement from the point of balancing a PUT. Specifically, we seek to reuse existing operator placement algorithms optimizing utility and augment them with privacy by reducing the search space for the operator placement to *trusted nodes* (cf. Figure 3). Prinseps relies on privacy action rules derived from the user's privacy policy applied with respect to the PPM to decide on the node's trustworthiness, like trusted sinks specified by AC-based PPMs (cf. Section 4.2), which limit query processing to, e.g., Bob's or his family member's devices.

Research Opportunity. Whilst we outline the idea of reducing the search space for the operator placement to address invasive queries, the question of *how to* derive information on nodes' trustworthiness from the action rules and encapsulate it in existing placement algorithms remains open. Thus, another research avenue is to explore factors for determining trusted nodes in a multilayered SPS, like (1) support of specific PPMs verifiable by SPS's users, (2) the capability of running PPMs inside TEEs, (3) trust scores of an SPS node given by its peers [22], and (4) levels of transparency, e.g., reliance on open architectures as well as voluntary certification.

5 CONCLUSION

In this work, we present our vision for Prinseps, enabling holistic privacy protection in multilayered stream processing systems (SPSs) with a focus on balancing a privacy-utility tradeoff (PUT). Prinseps utilizes critical privacy threats in multilayered SPSs that we identify in order to systematically select, customize, and apply state-of-the-art privacy-preserving mechanisms (PPMs) to address these critical threats while promoting the PUT. Furthermore, we provide research opportunities for the found privacy threats and PPMs tackling them, outlining how such PPMs can interact with core SPS mechanisms, like query rewriting and operator placement.

Mikhail Fomichev, Manisha Luthra, Maik Benndorf, Pratyush Agnihotri. No One Size (PPM) Fits All: Towards Privacy in Stream Processing Systems. In the Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS), pp. 61–67, ACM press

6 ACKNOWLEDGMENTS

We would like to thank Majid Lotfian Delouee, He Gu, Rana Tallal Javed, and Espen Volnes for the initial discussions on this paper; Christine Schäler and Martin Schäler for their input on Swellfish privacy as well as the reviewers for their useful input. This work has been co-funded by the Research Council of Norway as part of the project Parrot (311197), DFG Collaborative Research Center (CRC) 1053– MAKI and DFKI Darmstadt.

REFERENCES

[1] Sayan Biswas and Catuscia Palamidessi. 2022. PRIVIC: A Privacy-preserving Method for Incremental Collection of Location Data. *arXiv preprint arXiv:2206.10525* (2022).

[2] Cara Bloom. 2022. Privacy Threat Modeling. (2022).

[3] Lukas Burkhalter, Anwar Hithnawi, Alexander Viand, Hossein Shafagh, and Sylvia Ratnasamy. 2020. TimeCrypt: Encrypted Data Stream Processing at Scale with Cryptographic Access Control. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*. 835–850.

[4] Jianneng Cao, Barbara Carminati, Elena Ferrari, and Kian-Lee Tan. 2010. Castle: Continuously Anonymizing Data Streams. *IEEE Transactions on Dependable and Secure Computing* 8, 3 (2010), 337–352.

[5] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-adaptive Differentially Private Stream Processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1375–1388.

[6] Rahul Dwarakanath, Boris Koldehofe, Yashas Bharadwaj, The An Binh Nguyen, David Eyers, and Ralf Steinmetz. 2017. TrustCEP: Adopting a Trust-based Approach for Distributed Complex Event Processing. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 30–39.

[7] Ecenaz Erdemir, Pier Luigi Dragotti, and Deniz Gunduz. 2022. Active Privacy-Utility Trade-off Against Inference in Time-Series Data Sharing. *arXiv preprint arXiv:2202.05833* (2022).

[8] Ferdinando Fioretto and Pascal Van Hentenryck. 2019. OptStream: Releasing Time Series Privately. *Journal of Artificial Intelligence Research* 65 (2019), 423–456.

[9] He Gu, Thomas Plagemann, Maik Benndorf, Vera Goebel, and Boris Koldehofe. 2023. Differential Privacy for Protecting Private Patterns in Data Streams. *arXiv preprint arXiv:2305.06105* (2023).

[10] Kun Guo and Qishan Zhang. 2013. Fast Clustering-based Anonymization Approaches with Time Constraints for Data Streams. *Knowledge-Based Systems* 46 (2013), 95–108.

[11] Omid Hajihassnai, Omid Ardakanian, and Hamzeh Khazaei. 2021. ObscureNet: Learning Attribute-invariant Latent Representation for Anonymizing Sensor Data. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 40–52.

[12] IBM Corporation. 2023. Query Rewrite. https://www.ibm.com/docs/en/guardium/11.2?topic=protect-query-rewrite.

[13] Haojian Jin, Boyuan Guo, Rituparna Roychoudhury, Yaxing Yao, Swarun Kumar, Yuvraj Agarwal, and Jason I Hong. 2022. Exploring the Needs of Users for Supporting Privacy-Protective Behaviors in Smart Homes. In *CHI Conference on Human Factors in Computing Systems*. 1–19.

[14] Noah Johnson, Joseph P Near, Joseph M Hellerstein, and Dawn Song. 2020. Chorus: A Programming Framework for Building Scalable Differential Privacy Mechanisms. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 535–551.

[15] Manisha Luthra, Boris Koldehofe, Niels Danger, Pascal Weisenberger, Guido Salvaneschi, and Ioannis Stavrakakis. 2021. TCEP: Transitions in Operator Placement to Adapt to Dynamic Network Environments. *J. Comput. System Sci.* 122 (2021), 94–125.

[16] Matteo Nardelli, Valeria Cardellini, Vincenzo Grassi, and Francesco Lo Presti. 2019. Efficient Operator Placement for Distributed Data Stream Processing Applications. *IEEE Transactions on Parallel and Distributed Systems* 30, 8 (2019), 1753–1767.

[17] Emanuel Onica, Pascal Felber, Hugues Mercier, and Etienne Rivière. 2016. Confidentiality-preserving Publish/Subscribe: A Survey. *ACM computing surveys (CSUR)* 49, 2 (2016), 1–43.

[18] Saravana Murthy Palanisamy, Frank Dürr, Muhammad Adnan Tariq, and Kurt Rothermel. 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*. 40–51.

[19] Thomas Plagemann, Vera Goebel, Matthias Hollick, and Boris Koldehofe. 2022. Towards Privacy Engineering for Real-time Analytics in the Human-centered Internet of Things. *arXiv preprint arXiv:2210.16352* (2022).

[20] Do Le Quoc, Martin Beck, Pramod Bhatotia, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. 2017. PrivApprox: Privacy-Preserving Stream Analytics. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*. USENIX Association, 659–672.

[21] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *Proceedings of the 2022 International Conference on Management of Data*. 1064–1077.

[22] Mastooreh Salajegheh, Shashank Agrawal, Maliheh Shirvanian, Mihai Christodorescu, and Payman Mohassel. 2022. CoRA: Collaborative Risk-aware Authentication. *Cryptology ePrint Archive* (2022).

[23] Guido Salvaneschi, Mirko Köhler, Daniel Sokolowski, Philipp Haller, Sebastian Erdweg, and Mira Mezini. 2019. Language-integrated Privacy-aware Distributed Queries. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 167–1.

[24] Gianluca Scopelliti, Sepideh Pouyanrad, Job Noorman, Fritz Alder, Christoph Baumann, Frank Piessens, and Jan Tobias Mühlberg. 2022. End-to-End Security for Distributed Event-Driven Enclave Applications on Heterogeneous TEEs. *arXiv preprint arXiv:2206.01041* (2022).

[25] Christoph Stach, Clémentine Gritti, and Bernhard Mitschang. 2020. Bringing Privacy Control Back to Citizens: DISPEL—A Distributed Privacy Management Platform for the Internet of Things. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 1272–1279.

[26] Timo Sztyler and Heiner Stuckenschmidt. 2016. On-body Localization of Wearable Devices: An Investigation of Position-aware Activity Recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–9.

[27] Christine Tex, Martin Schäler, and Klemens Böhm. 2022. Swellfish Privacy: Supporting Time-dependent Relevance for Continuous Differential Privacy. *Information Systems* (2022), 102079.

[28] Xin Yang and Omid Ardakanian. 2022. Blinder: End-to-end Privacy Protection in Sensing Systems via Personalized Federated Learning. *arXiv preprint arXiv:2209.12046* (2022).

[29] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, Haralampos Gavriilidis, Dimitrios Giouroukis, Philipp M Grulich, Sebastian Breß, Jonas Traub, and Volker Markl. 2020. The NebulaStream Platform for Data and Application Management in the Internet of Things. In *10th Conference on Innovative Data Systems Research, CIDR 2020*.