Differentially private stream processing for the semantic web

Dell'Aglio, Daniele ; Bernstein, Abraham

Abstract: Data often contains sensitive information, which poses a major obstacle to publishing it. Some suggest to obfuscate the data or only releasing some data statistics. These approaches have, however, been shown to provide insufficient safeguards against de-anonymisation. Recently, differential privacy (DP), an approach that injects noise into the query answers to provide statistical privacy guarantees, has emerged as a solution to release sensitive data. This study investigates how to continuously release privacy-preserving histograms (or distributions) from online streams of sensitive data by combining DP and semantic web technologies. We focus on distributions, as they are the basis for many analytic applications. Specifically, we propose SihlQL, a query language that processes RDF streams in a privacy-preserving fashion. SihlQL builds on top of SPARQL and the w-event DP framework. We show how some peculiarities of w-event privacy constrain the expressiveness of SihlQL queries. Addressing these constraints, we propose an extension of w-event privacy that provides answers to a larger class of queries while preserving their privacy. To evaluate SihlQL, we implemented a prototype engine that compiles queries to Apache Flink topologies and studied its privacy properties using real-world data from an IPTV provider and an online e-commerce web site.

DOI: https://doi.org/10.1145/3366423.3380265

Differentially Private Stream Processing for the Semantic Web

Daniele Dell'Aglio
dellaglio@ifi.uzh.ch
University of Zurich
Zurich, Switzerland

Abraham Bernstein
bernstein@ifi.uzh.ch
University of Zurich
Zurich, Switzerland

ABSTRACT

Data often contains sensitive information, which poses a major obstacle to publishing it. Some suggest to obfuscate the data or only releasing some data statistics. These approaches have, however, been shown to provide insufficient safeguards against de-anonymisation. Recently, differential privacy (DP), an approach that injects noise into the query answers to provide statistical privacy guarantees, has emerged as a solution to release sensitive data. This study investigates how to continuously release privacy-preserving histograms (or distributions) from online streams of sensitive data by combining DP and semantic web technologies. We focus on distributions, as they are the basis for many analytic applications. Specifically, we propose *SihlQL*, a query language that processes RDF streams in a privacy-preserving fashion. *SihlQL* builds on top of SPARQL and the w-event DP framework. We show how some peculiarities of w-event privacy constrain the expressiveness of *SihlQL* queries. Addressing these constraints, we propose an extension of w-event privacy that provides answers to a larger class of queries while preserving their privacy. To evaluate *SihlQL*, we implemented a prototype engine that compiles queries to Apache Flink topologies and studied its privacy properties using real-world data from an IPTV provider and an online e-commerce web site.

ACM Reference Format:
Daniele Dell'Aglio and Abraham Bernstein. 2020. Differentially Private Stream Processing for the Semantic Web. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380265

1 INTRODUCTION

Companies, public administrations, and individuals show an active interest in sharing their data on the web. The linked open government data movement is a successful example of this trend [21], which has been joined by several companies, such as BBC and New York Times. While there is potential value in sharing data about citizens or customers, there are critical privacy-related risks that must be taken into account. In the past, data owners used obfuscation and anonymisation techniques to share datasets, but these solutions led to scandals like the Netflix challenge [20] and the Massachusetts hospital [24] ones. Record linkage is a common technique to crack those techniques: by discovering links between the anonymised dataset and one, where users are known, researchers have shown that user re-identification is possible. A solution emerged in mid-2000s when Dwork proposed differential privacy [10]. The intuition behind differential privacy is to introduce noise into query answers, such that it becomes hard to state if the data contains a specific user. Differential privacy has been intensively studied in the last decade and today is used in different companies, such as Google and Apple [11, 25], to let their data scientists analyse the data collected from their users in a privacy-preserving manner.

To date, data publication has mainly focused on static datasets, managing their evolution through archiving [12]. Data publication is more complex in the dynamic setting, where responsiveness is usually a key requirement. Some suggest managing updates as data streams, publishing changes as they happen [5]. Similarly, privacy in streams introduces new challenges related to the data dynamics, which may be used to infer private information.

This study investigates how to continuously publish data extracted from private data streams containing user-related information to the web of data in a privacy-preserving manner. Consider the following example scenario: a company processes a private stream carrying information from an IPTV platform containing viewership information. The stream describes what users watch, registering when users switch channels. The company enriches the stream with data from private and public data sources, and wants to publish their analyses in the web to showcase their analytic capabilities. The company also wants to be compliant with privacy regulations and avoid scandals like the ones mentioned above. It is worth noting that there can be privacy leaks when only releasing statistics, since they may highlight the behaviour of outliers which could be re-identified by exploiting external knowledge.

As a solution, we propose *SihlQL*, a query language to perform data analytic tasks over streams while preserving privacy. *SihlQL* focuses on the creation of histograms (or distributions), as they provide the foundation for many analytic queries in applications such as data warehousing, OLAP and business analytics, as well as plenty of machine learning algorithms such as decision tree learners or naïve Bayes. *SihlQL* extends SPARQL to support the processing of data streams and narrows SPARQL by introducing constraints to guarantee that results are compliant with differential privacy. As such, the main contributions of this article are:

- *SihlQL*: a continuous SPARQL-based query language for RDF streams that integrates differential privacy methods;
- a new differential privacy mechanism that extends the expressive power of w-event privacy [14] and its suitability for SPARQL by strategically dropping histogram bins; and
- a *SihlQL* execution engine prototype built on top of Apache Flink that rewrites *SihlQL* queries as Flink topologies.

Next, Section 2 describes related work and introduces the basic notions of differential privacy and RDF stream processing. Section 3 presents the design of *SihlQL*, which is tailored to enable w-event

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380265

privacy. Section 4 introduces our extension of w-event privacy—the bin removal mechanism—and integrates it to the w-event privacy framework and *SihlQL*. Section 5 analyses *SihlQL* with experiments over two real-world data sets. Section 6 concludes the article with final remarks and future work.

2 BACKGROUND AND RELATED RESEARCH

This section introduces differential privacy with a specific focus on w-event privacy, which is one of the building blocks of this study. Next, we review the main concepts of RDF stream processing and privacy in the semantic web context.

Differential privacy. Differential privacy (DP) [10] introduces noise in query answers to protect the presence of users in the dataset without significantly changing the result. Central to DP is the notion of *neighbouring datasets*: two datasets are neighbours if they differ in one record. Given two neighbouring datasets D and D', a query Q (or mechanism), DP imposes plausible deniability for each possible answer E by comparing the probability P that the answer can be found when running the query on D and D' against a threshold e^ϵ, i.e.

$$P\left(Q(D) \in E\right) \leq e^\epsilon \cdot P(Q(D') \in E) + \delta. \quad (1)$$

$\epsilon \geq 0$ and $\delta \geq 0$ are two parameters that regulate the differential privacy mechanisms. ϵ is called *privacy budget* (shortly budget) and trades off utility and privacy. When $\delta = 0$, we have ϵ-*indistinguishability*, i.e. the ratio between the two probabilities in Eq. 1 is bound by e^ϵ. This is an ideal case since it guarantees that any output produced by $Q(D)$ is likely to be generated by evaluating Q on every neighbour dataset D'. Without this property, there are outputs from $Q(D)$ which are more (or less) likely to be generated by Q on some neighbour dataset D'. When $\delta > 0$, there is a probability $1 - \delta$ that the output of Q is not ϵ-indistinguishable.

DP is implemented through mechanisms that execute the query Q and add some noise to the result. Such noise is usually sampled from a Laplace distribution, calibrated to ϵ such that Eq. 1 holds. This led to the development of a large set of mechanisms for operations ranging from simple (e.g., sum and count [10]) to more complex (e.g. deep learning [1]). Looking at interfaces for differentially private querying, McSherry proposes PINQ [19], an extension for Microsoft LINQ for DP. PINQ is a programmable API inspired by SQL to let the user specify how to process data stored in static datasets.

Data streams are different than static databases. Streams contain continuous updates, hence query answers also become continuous. The introduction of streams requires a definition of neighbouring streams (rather than databases).

In [9], Dwork introduces event- and user-level privacy. *Event-level privacy* defines two streams as neighbours if they differ in one stream item. Event-level privacy is a straightforward extension of neighbour datasets and led to the creation of mechanisms for computing different types of queries, such as count and histograms [6, 9]. The main drawback is that ensuring plausible deniability w.r.t. streams changing in only one item exposes privacy leaks in cases where such streams contain multiple stream items describing the same subject, such as financial transactions or location streams. *User-level privacy* overcomes this drawback by defining two streams as neighbours if they differ for one user (or object to be protected). An issue with this definition is that it may require a high injection of noise in the query results, destroying the utility of the answers. This is because two neighbour streams can be completely different. E.g., if a stream contains only one user, the empty stream would be a neighbour, and it requires high noise injection to make the answers over them indistinguishable. To the best of our knowledge, *SihlQL* is the first declarative query language proposed for differentially private queries on streams.

w-event privacy. Kellaris et al. propose w-event privacy [14] to overcome the limitations of the event- and user-level privacy. It defines two streams as neighbours if they differ at most by one item in a window of size w which slides along the stream, and guarantees that no window uses more budget than ϵ.

Let $\mathcal{D} = (D_1, D_2, \ldots)$ be a stream of datasets. Each dataset D_i contains tuples (s, a), where s is the sensitive value to be protected (e.g., user identities) and a is the value to analyse (e.g., TV channels); two tuples in D_i cannot share the value s, and s can appear in multiple datasets of $\mathcal{D}$. The task to be executed is represented by a query Q that processes each individual dataset D_i in $\mathcal{D}$ and outputs an answer $\mathbf{o_i}$, which is appended to the output stream $(\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_i}, \ldots)$.

Kellaris et al. propose different algorithms and variations. In this study, we focus on the Budget Distribution (BD) scheme, but our contribution can also be applied to the other schemes they propose. In the beginning, BD allocates a budget ϵ/w for the processing of every dataset in $\mathcal{D}$, ensuring that in w consecutive evaluations the maximum budget used is ϵ. The idea of the BD scheme is that the output stream may contain empty (or *null*) answers: the current $\mathbf{o_i}$ can be skipped (i.e., can be null) when it does not significantly differ from the last non-null output $\mathbf{o_l}$. When an answer is null, the budget can also be *saved* to process the next dataset in the stream. The application of BD to the i-th item of $\mathcal{D}$ is presented in Algorithm 1. The execution of Q on D_i produces as answer a vector of values $\mathbf{c_i}$ (Line 1). Next, BD computes the difference between $\mathbf{c_i}$ and the last non-null output $\mathbf{o_l}$, adds to it noise drawn from a Laplacian distribution and stores it in a variable *dis* (Lines 3-4). This step uses half of the privacy budget ϵ allocated for the processing of D_i (i.e. $\epsilon/(2 \cdot w)$), as shown in Line 3. *dis* is compared with $\lambda_{i,2}$ (Line 7) to determine if $\mathbf{o_i}$ should be null or not. $\lambda_{i,2}$ is half of the remaining budget ϵ_{rm} (Line 6), which is computed by subtracting from $\epsilon/2$ the budget used in the previous w iterations (Line 5). If *dis* is greater than $\lambda_{i,2}$, noise is added to $\mathbf{c_i}$ (calibrated by $\lambda_{i,2}$) and the algorithm releases a new answer, paying a price of $\epsilon_{rm}/2$ (Line 7). Otherwise, the privacy budget is preserved, the algorithm returns a null answer and the data analyst will assume that the last non-null release $\mathbf{o_l}$ is a good approximation of $\mathbf{o_i}$.

w-event privacy is one of the most sophisticated state of the art solutions to enable DP in stream processing. Compared to simpler alternatives, this framework usually achieves better utility by using the same amount of privacy budget. w-event privacy is presented as a set of algorithms, but the authors do not propose a query language on top of it. *SihlQL* is, therefore, the first declarative query language built on top of this framework. This version of *SihlQL* focuses on a specific operation—histogram computation—but it can be extended to integrate a wider set of operators based on w-event privacy.

Algorithm 1 Pseudocode of the w-event privacy BD scheme [14]

1: $\mathbf{c_i} \leftarrow Q(D_i)$
2: Identify the last non-null release $\mathbf{o_l}$ from $(\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_{i-1}})$
3: $dis \leftarrow \frac{1}{d}\sum_{j=1}^{d} |\mathbf{o_l}[j] - \mathbf{c_i}[j]|$; $\lambda_{i,1} \leftarrow (2 \cdot w)/(\epsilon \cdot d)$
4: $dis \leftarrow dis + Lap(\lambda_{i,1})$
5: $\epsilon_{rm} \leftarrow \epsilon/2 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2}$
6: $\lambda_{i,2} \leftarrow 2/\epsilon_{rm}$
7: **if** $dis > \lambda_{i,2}$ **then** $\epsilon_{i,2} \leftarrow \epsilon_{rm}/2$; **return** $\mathbf{o_i} \leftarrow \mathbf{c_i} + \langle Lap(\lambda_{i,2})\rangle^d$
8: **else** $\epsilon_{i,2} \leftarrow 0$; **return** $\mathbf{o_i} \leftarrow null$

RDF stream processing. RDF stream processing studies how to extend the semantic web stack with models to capture streams and process them [8]. An RDF stream $\mathcal{S}$ is a sequence of time-annotated RDF graphs $((G_1, t_1), (G_2, t_2), \ldots)$, where G_i denotes the i-th RDF graph and t_i is its temporal annotation. We assume that the temporal annotation is a time instant, and that graphs are ordered by time, e.g. as in [4, 16, 17]. For example, let $\mathcal{S}_{TV}$ be an RDF stream containing information about which channels users are currently watching. At time t_i, G_i reports on the current state of viewers with RDF triples (u :isWatching c), where u and c are IRIs identifying respectively users and channels.

Query languages for processing RDF streams are extensions of SPARQL [13]. A SPARQL query is defined as a tuple (E, DS, QF). E is a SPARQL algebra expression, which is composed by combining graph patterns through algebraic operators, e.g., joins ($\bowtie$), left joins ($⟕$), unions ($\cup$), and groups (γ). Let I be the set of IRIs; $DS = \{(def, G), (u_1, G_1), \ldots, (u_k, G_k)\}$ denotes an RDF dataset, where def identifies the *default* RDF graph G, and $u_i \in I$ ($i \leq k$) denotes the *named* graph G_i. A dataset contains one default graph and zero or more named graphs, i.e. $k \geq 0$. QF is a query form among SELECT, CONSTRUCT, ASK and DESCRIBE. To execute a SPARQL query, algebra operators are replaced by physical operators that implement the evaluation semantics of the relative operators. Physical operators consume and generate bags Ω of solution mappings μ—partial functions associating variables to RDF terms (the set of IRIs, literals and blank nodes). For example, the SPARQL query in Listing 1 computes histograms by processing data from a graph G_{TV}. The query can be represented as:

$$q = (E = \gamma_{?channel, COUNT(*)}(BGP),\\ DS = \{(def, :G_{TV})\},\\ QF = \text{SELECT}).$$

E describes a sequence of two operations: (1) a basic graph pattern (Line 4) produces solution mappings μ that associate two variables ?*user* and ?*channel* to RDF terms and (2) a grouping operator creates groups of mappings sharing the same ?*channel* values (Line 5), and computes the size of each group (Line 3). Continuous extensions of SPARQL exploit time annotations to perform operations over streams, including window-based operations (e.g. C-SPARQL [4] and CQELS [16]) and event pattern matching (e.g. EP-SPARQL [2] and DOTR [17]). They are built by extending DS to include streams (in addition to graphs), and the set of operators to compose E with stream-related operators [7]. Moreover, they extend the evaluation semantics to move from one-time to continuous. The result of those

```
1 PREFIX : <http://example.com/>
2 FROM :G_TV
3 SELECT ?channel (COUNT(*) AS ?viewers)
4 WHERE{ ?user :isWatching ?channel }
5 GROUP BY ?channel
```

Listing 1: Histogram computation in SPARQL

continuous query evaluations is a stream which contains the answers that are computed over time. These solutions focused on creating languages with a large number of operations, to fulfil a large number of tasks. In this study, we take a different perspective focusing on the definition of a *differentially private language for processing RDF streams*. Intuitively, there are queries which are intrinsically harder than others to *protect*. For example, select-project-join queries reveal more information about the underlying data than aggregation queries and are, consequently, harder to protect. Therefore, a differentially-private query language may sacrifice expressive power for the sake of preserving privacy.

Privacy in the semantic web. Kirrane et al. [15] provide an overview of the problems and solutions of privacy, security and associated policies in the context of the semantic web technologies. Whilst they found various papers tackling the issue of privacy and we are aware of some papers exploring the adaption of DP in the static case [23], to the best of our knowledge no publication considered the application of differential privacy to semantic web data streams.

3 A QUERY LANGUAGE FOR W-EVENT PRIVACY

In this section, we introduce *SihlQL*, a query language relaying on and extending the notion of w-event privacy. We first present a set of constraints derived from w-event privacy and then we describe how SPARQL can be extended to fulfil them.

3.1 w-event privacy constraints

Defining a SPARQL-based query language for processing RDF streams while preserving w-event privacy requires to take into account the constraints the latter introduces. Looking at the input data, w-event privacy assumes that: (C1) there is *one* input stream, (C2) each stream item contains *all the data required* to compute the relative answer, and (C3) in each stream item, the value to be protected appears at most in one tuple.

In addition, w-event privacy imposes two assumptions on the parameter d (Line 3 in Algorithm 1), which describes the number of answer groups (e.g. bins) returned: (C4) d is a constant determined ex-ante during query definition time, and (C5) at each time instant, the size of $\mathbf{c_i}$ is d. In the task we want to achieve—histogram computation—d is the number of bins to be computed.

3.2 Dataset

Constraints C1, C2, and C3 drive the modelling of the *SihlQL* dataset. A *SihlQL* dataset is composed by one RDF stream $\mathcal{S}$ and zero or more named RDF graphs G, i.e.:

$$DS_{SihlQL} = \{(def, \mathcal{S}), (u_1, G_1), \ldots, (u_k, G_k)\}, k \geq 0,$$

where def denotes the data stream and $u_i \in I$ the named graphs ($i \leq k$). The *SihlQL* dataset intrinsically satisfies C1. $\mathcal{S}$ takes the role of the default graph in SPARQL and it captures the idea that

the *default* data source to be processed by *SihlQL* is the stream. Comparing *SihlQL* with other query languages for RDF streams, it differs from CQELS, since CQELS sets streams as named elements. *SihlQL* is similar to C-SPARQL, where all the input streams are merged into a default stream. This is a viable solution also in *SihlQL*, but introduces the need to assess that C3 holds after the merge.

C2 requires that the stream conforms to the CQL's RStream definition [3]: each stream item reports a description of the events happening at the relative time instant. We keep this assumption for the sake of clarity, but it can be relaxed to include streams with events annotated with the start and end time instants.

C3 also implies that one knows ex-ante which part of the data is sensitive. Hence, we assume that the query is submitted by a reliable user such as the data curator. A possible way to overcome this limitation is to annotate the data to describe its level of privacy. For example, if the query contains the triple pattern at Line 3 of Listing 1 and the domain of $:isWatching$ is declared as sensitive, queries which projects or groups $?user$ will be rejected. This can be implemented by designing meta-level annotations over the data schema, similar to the approach that Zhao et al. propose to describe data dynamics [26].

3.3 Algebra operators

SihlQL supports the usual algebra operators of SPARQL [13]: selection, projection, join, left join, grouping and aggregations, in addition to basic graph patterns and GRAPH clauses. Differently from other extensions for querying streams, *SihlQL* does not introduce new operators. Using stream item-aggregator operators like sliding windows may affect the privacy of the answer since the data from the same stream item would be processed multiple times. This causes a potential loss of w-event privacy guarantees. Having sliding windows may also break the condition C3 when a window captures two stream items containing the same sensitive value.

In addition, to preserve the conditions C4 and C5 set by w-event privacy, *SihlQL* limits the way on which E_{SihlQL} (i.e. the algebra expressions) can be composed. C5 sets the number of bins as constant over time. This happens when either the stream has information about empty bins or the bin set is available from some *background data*. When streams describe events, however, it often happens that no information is available about entities not involved in them. E.g. if no one is watching the TV channel c, c will not appear in $\mathcal{S}_{TV}$. Having information about the bins stored in some background data is a common use case, and it follows that *SihlQL* should support the join of static background data and the input stream. This join must be a *left join* to ensure that empty bins are contained in the result. An inner join would not produce empty bins. For example, if at time t a channel c has no viewers, the relative stream item will not contain any triple describing it, and the inner join will produce zero solution mappings about c.

It follows that the *SihlQL* WHERE clause should contain a GRAPH and an OPTIONAL clause as in Listing 2: channels are loaded from background data $:G_{channels}$ and the combination between streaming and background data is regulated by an OPTIONAL clause. This is a minimal structure, and *SihlQL* queries support more complex WHERE clauses, where graph patterns can be added to enrich, filter or extend the solution mappings. For example, it is possible to put

```
1 PREFIX : <http://example.com/>
2 ENABLE PRIVACY EPSILON 0.1 W 3
3 SELECT ?channel (COUNT(*) AS ?viewers)
4 FROM STREAM :S_TV
5 FROM STATIC :G_channels
6 WHERE {
7 OPTIONAL { ?user :isWatching ?channel }
8 GRAPH :G_channels { ?channel a :TVChannel }
9 } GROUP BY ?channel
```

Listing 2: Histogram computation in *SihlQL*

conditions to select only the English-speaking TV channels or to compute histograms of viewers with age between 18 and 35.

Note that the required OPTIONAL clause is a strong limitation when defining queries. In Section 4 we introduce an extension to the notion of w-event privacy, which allows us to relax this constraint of fixed bin set size and consequently remove the need of the OPTIONAL and GRAPH clauses.

It is worth mentioning that even if some constraints can be checked at query compile time (e.g. the presence of OPTIONAL and GRAPH clause), other constraints, such as C3 or C5, cannot and can be only detected at evaluation time.

3.4 Evaluation semantics and query forms.

The continuous and instantaneous evaluation semantics define respectively *when* and *how* the query is evaluated [7]. We designed *SihlQL* to compute a new output every time new data is available on the input stream. Therefore, there is an instantaneous evaluation at each time instant t_i such that (G_i, t_i) is a stream item of $\mathcal{S}$. This makes the *SihlQL* continuous evaluation semantics ideal to process one stream item at a time, similarly to CQELS and EP-SPARQL semantics.

The instantaneous semantics of *SihlQL* is analogous to SPARQL's. When the query is evaluated at time t_i, the dataset is transformed to $DS^{t_i}_{SihlQL} = \{(def, G_i), (u_k, G_k)\}$ where (G_i, t_i) is an item of the input stream $\mathcal{S}$ and $k >= 0$. Hence, a SPARQL dataset is built by setting the stream item associated to the evaluation time instant as default graph. Since *SihlQL* does not introduce new operators and it supports SELECT and CONSTRUCT as query forms, any *SihlQL* expression E_{SihlQL} leads to a SPARQL query $(DS^{t_i}_{SihlQL}, E_{SihlQL}, QF)$.

The instantaneous evaluation of a SELECT query produces solution bindings, while the instantaneous evaluation of CONSTRUCT queries produce RDF graphs. Instantaneous answers are annotated with the evaluation time instant and become items of the output stream: a stream of time-annotated bindings in case of SELECT queries and an RDF stream in case of CONSTRUCT queries.

3.5 Privacy considerations

To complete the description of *SihlQL*, we need to introduce the privacy aspects. In Section 2 we explained that w-event privacy requires two parameters, ϵ and w. *SihlQL* introduces a PRIVACY clause to enable privacy and to set such parameters. An example is at Line 2 in Listing 2, where ENABLE PRIVACY indicates that the query should consider differential privacy. The second part of the clause declares that $\epsilon = 0.1$ and $w = 3$. This design choice to let the user set the privacy parameters is similar to the one in

PINQ [19], and it assumes that the query writer is a trusted user and will set fair parameter values. Additionally, the system could set a maximum privacy budget ϵ_{max}, and when registered queries use all the budget, no more queries are accepted.

w-event privacy is implemented as a physical operator that can be used when the query compiles the algebra expression. This physical operator implements Algorithm 1 and performs a left join between two sets of mappings, it groups and aggregates them. In other words, it implements an algorithm that evaluates

$$\gamma_{?bin,COUNT(*)}(\Omega_b \bowtie_{?bin} \Omega_h), \tag{2}$$

where Ω_b is the set of solution mappings including the list of bins, Ω_h is the set of solution mappings with the data to compute the bin heights, and $?bin$ is the common variable between mappings of Ω_b and Ω_h. Taking as an example the query in Listing 2, Ω_b and Ω_h are the bags produced by the evaluation of the GRAPH and OPTIONAL clauses (Lines 8 and 7 resp.), and the common variable between the mappings of the two bags is $?channel$.

4 EXTENDING SIHLQL EXPRESSIVENESS

The main limitation of the query language we designed in Section 3 is the relation between the w-event privacy algorithm and the algebra expression it implements (Eq. 2). It implies a set of constraints on both the data and the algebra expression that the query must satisfy to ensure that the privacy-preserving computation occurs (correctly). We observe that Eq. 2 is a direct consequence of the constraints C4 and C5: the list of bins should be known a priori, and every bin must appear in every non-null answer of the output.

The major obstacle to relaxing these constraints relates to the empty bins: the privacy-preserving evaluation of the histograms adds noise to every bin (Line 7 in Algorithm 1) independently of its size. If we replace the left join in Eq. 2 with

$$\gamma_{?bin,COUNT(*)}(\Omega_b \bowtie_{?bin} \Omega_h) \tag{3}$$

then the solution will contain only non-empty bins, potentially leading to privacy leaks. For example, if the query isolates one user of $\mathcal{S}_{TV}$ then the result would contain only one non-empty TV channel bin and evaluating Eq. 3 would unveil which channel the user is watching, even if the bin height is obfuscated with noise. Eq. 2 avoids this situation by always returning all bins, whose magnitude is obfuscated with noise. Hence, it is hard to distinguish between empty bins and bins with one user. In this section, we introduce an original extension of w-event privacy that addresses this issue via a bin-removal mechanism.

4.1 The bin removal mechanism.

Our intuition to overcome this issue is to remove both empty and almost empty bins. As a consequence, malicious data analysts would not know if bins are missing because they are empty or almost empty. Setting a threshold h_{th} such that bins containing less than h_{th} entries are automatically removed would not solve the problem, as it would allow exploiting such information to violate privacy when the threshold is set to 0. We therefore design the bin removal mechanism $\mathcal{R}$, defined as follows.

Definition 4.1. Let $\mathcal{R}$ be a mechanism that takes as an input a bin height and returns a boolean value: *true* to keep the bin or *false* to remove it. The output is a sample from a Bernoulli distribution $Be(p)$, where p is defined as

$$p = \frac{1}{1 + e^{-k(\epsilon)\cdot(h-h_0)}}, \tag{4}$$

having $k(\epsilon)$ defined as a function proportional to ϵ.

$\mathcal{R}$ is a stochastic process: the fewer elements a bin contains, the higher the probability that it is removed. This behaviour can be modelled with a Bernoulli distribution, where the argument p is computed via a logistic function—a sigmoid function with parameters to control the maximum value, the steepness, and the mid-value position of the curve. The probability p that a bin of height h is kept is $\frac{1}{1+e^{-k\cdot(h-h_0)}}$, where $h_0 \geq 0$ is the height which sets the probability to 0.5 and $k \geq 0$ is the steepness of the curve. k impacts the privacy: the higher this value, the higher the steepness, making it easier to infer the actual bin height. We, therefore, relate k to the privacy budget ϵ by setting the former as a function of the latter. As the privacy budget decreases, the steepness of the curve decreases and the removal probability of a bin containing h items is closer to the removal probability of the neighbouring bins (i.e. containing $h \pm 1$ items), leading to more privacy. Therefore, $k(\epsilon)$ must be proportional to ϵ.

The following theorem defines the DP guarantees of $\mathcal{R}$.

Theorem 1. *$\mathcal{R}$ is (ϵ,δ)-differentially private, having δ defined as:*

$$\delta = \max\{0, d(\bar{x})\}, \tag{5}$$

where

$$d(\bar{x}) = \frac{1}{1 + e^{-k(\epsilon)\cdot(\bar{x}-h_0)}} - \frac{e^{\epsilon}}{1 + e^{-k(\epsilon)\cdot(\bar{x}-1-h_0)}}. \tag{6}$$

If $d(x)$ is not monotonically decreasing, $\bar{x}$ is the value that maximises $d(x)$, i.e.

$$\bar{x} = \frac{1}{k} \cdot \ln\left(\frac{-r + \sqrt{r^2 - 4\cdot s\cdot q}}{2\cdot s}\right) \tag{7}$$

where

$$s = e^{k(\epsilon)\cdot h_0} \cdot (e^{\epsilon + k(\epsilon)} - 1) \tag{8}$$

$$r = k(\epsilon) \cdot e^{2\cdot k(\epsilon)\cdot h_0 + k(\epsilon)} \cdot (e^{\epsilon} - 1) \tag{9}$$

$$q = e^{3\cdot k(\epsilon)\cdot h_0 + k(\epsilon)} \cdot (e^{\epsilon} - e^{k(\epsilon)}) \tag{10}$$

Proof. Let b_1 and b_2 be two neighbouring bins with x and y items, i.e. $|y - x| = 1$. Differential privacy imposes plausible deniability for every possible outcome: the probabilities that $\mathcal{R}$ keeps b_1 and b_2 must be *close*, as well as the probabilities that $\mathcal{R}$ removes them. Following the definition of (ϵ, δ)-DP in Eq. 1, the two cases are respectively formally modelled as

$$P(\mathcal{R}(b_1) = \text{kept}) \leq e^{\epsilon} \cdot P(\mathcal{R}(b_2) = \text{kept}) + \delta \tag{11}$$

$$P(\mathcal{R}(b_1) = \neg\text{kept}) \leq e^{\epsilon} \cdot P(\mathcal{R}(b_2) = \neg\text{kept}) + \delta. \tag{12}$$

We first study Eq. 11, which can be rewritten as:

$$\frac{1}{1 + e^{-k(\epsilon)\cdot(x-h_0)}} \leq \frac{e^{\epsilon}}{1 + e^{-k(\epsilon)\cdot(y-h_0)}} + \delta. \tag{13}$$

Let $f_{b_1}(x)$ and $f_{b_2}(y)$ be $\frac{1}{1+e^{-k(\epsilon)\cdot(x-h_0)}}$ and $\frac{e^{\epsilon}}{1+e^{-k(\epsilon)\cdot(y-h_0)}}$ respectively. When b_2 is larger than b_1, i.e. $y = x + 1$, $f_{b_1}(x)$ and $f_{b_2}(y)$

never intersect. We can show this by rewriting Eq. 13 with $\delta = 0$ and $y = x + 1$:

$$e^{-k(\epsilon)\cdot x - k(\epsilon) + k(\epsilon)\cdot h_0} - e^{\epsilon - k(\epsilon)\cdot x + k(\epsilon)\cdot h_0} \leq e^{\epsilon} - 1 \quad (14)$$

$$\frac{e^{-k(\epsilon)+k(\epsilon)\cdot h_0} - e^{\epsilon + k(\epsilon)\cdot h_0}}{e^{k(\epsilon)\cdot x}} \leq e^{\epsilon} - 1 \quad (15)$$

$$\frac{e^{-k(\epsilon)+k(\epsilon)\cdot h_0} - e^{\epsilon + k(\epsilon)\cdot h_0}}{e^{\epsilon} - 1} \leq e^{k(\epsilon)\cdot x} \quad (16)$$

Eq. 16 holds when $\epsilon, k(\epsilon) > 0$. The right term is positive; while the left one is negative: the numerator is negative (ϵ and $k(\epsilon)$ are positive, so $e^{\epsilon + k(\epsilon)\cdot h_0} > e^{-k(\epsilon)+k(\epsilon)\cdot h_0}$) and the denominator is positive.

When b_2 is smaller than b_1, i.e. $y = x - 1$, $f_{b_1}(x)$ and $f_{b_2}(x-1)$ may or may not intersect. When the two functions intersect, there is an area where $f_{b_1}(x)$ is bigger than $f_{b_2}(y)$. We need to set δ to a value that let the two logistic functions touch in exactly one point to avoid such area, and consequently let Eq. 13 hold.

Formally, let $d(x)$ be the function $f_{b_1}(x) - f_{b_2}(x-1)$. The maximum value of $d(x)$ is negative when $f_{b_1}(x)$ and $f_{b_2}(x-1)$ do not intersect; 0 or positive otherwise. It follows that $\delta = \max\{0, \max\{d(x)\}\}$.

Let $\bar{x}$ be the value that maximises $d(x)$. We can find $\bar{x}$ by solving the equation $\frac{\partial d}{\partial x} = 0$, i.e.

$$\frac{k(\epsilon)\cdot e^{-k(\epsilon)\cdot(x-h_0)}}{(1+e^{-k(\epsilon)\cdot(x-h_0)})^2} - \frac{k(\epsilon)\cdot e^{\epsilon - k(\epsilon)\cdot(x-1-h_0)}}{(1+e^{-k(\epsilon)\cdot(x-1-h_0)})^2} = 0 \quad (17)$$

By expanding this equation, we obtain:

$$e^{\epsilon - k(\epsilon)\cdot x + k(\epsilon) + k(\epsilon)\cdot h_0} - e^{-k(\epsilon)\cdot x + k(\epsilon)\cdot h_0} +$$
$$k(\epsilon)\cdot(e^{\epsilon - 2\cdot k(\epsilon)\cdot x + k(\epsilon) + 2\cdot k(\epsilon)\cdot h_0} - e^{-2\cdot k(\epsilon)\cdot x + k(\epsilon) + 2\cdot k(\epsilon)\cdot h_0}) +$$
$$e^{\epsilon - 3\cdot k(\epsilon)\cdot x + k(\epsilon) + 3\cdot k(\epsilon)\cdot h_0} - e^{-3\cdot k(\epsilon)\cdot x + 2\cdot k(\epsilon) + 3\cdot k(\epsilon)\cdot h_0} = 0$$

By multiplying both sides for $e^{3\cdot k(\epsilon)\cdot x}$, we obtain:

$$s\cdot e^{2\cdot k(\epsilon)\cdot x} + r\cdot e^{k(\epsilon)\cdot x} + q = 0$$

where s, r and q are defined as in Eq. 8, 9 and 10. We can replace $e^{k(\epsilon)\cdot x}$ with a variable $\hat{x}$, resulting in a second grade equation: $s\cdot\hat{x}^2 + p\cdot\hat{x} + q$, which has solutions

$$\hat{x} = \frac{-r \pm \sqrt{r^2 - 4\cdot s\cdot q}}{2\cdot s}. \quad (18)$$

Since $\hat{x} = e^{k(\epsilon)\cdot x}$, it follows that $x = \frac{1}{k(\epsilon)}\cdot\ln(\hat{x})$. We can observe that s and r are non-negative since $\epsilon, k(\epsilon) > 0$, i.e. $-\frac{r}{2s}$ is negative. q is 0 or negative when $\epsilon \geq k(\epsilon)$, positive otherwise. That means $\hat{x}$ does not exist when q is big enough to let $\Delta = r^2 - 4\cdot s\cdot q$ be negative, i.e. $f_{b_1}(x)$ and $f_{b_2}(x-1)$ do not intersect and $d(x)$ is monotonically decreasing. When q has a value such that $\hat{x}$ admits solutions, $\hat{x}$ has at least one negative or 0 solution, given by $\frac{-r-\sqrt{\Delta}}{2\cdot s}$. The sign of the other solution of $\hat{x}$, i.e. $\frac{-r+\sqrt{\Delta}}{2\cdot s}$ can be either positive (when $r < \Delta$), 0 (when $r = \Delta$), or negative (when $r > \Delta$). We are interested in the latter case, which guarantees the existence of $\bar{x}$. That means, the only admissible solution $\bar{x}$ is $\frac{1}{k}\cdot\ln(\frac{-r+\sqrt{p^2-4\cdot s\cdot q}}{2\cdot s})$.

To summarise, $d(\bar{x})$ is the maximum value of $d(x)$. When $d(\bar{x})$ is positive, by setting $\delta = d(\bar{x})$ we guarantee that the two logistic functions $f_{b_1}(x)$ and $f_{b_2}(x-1)$ do not intersect, i.e., Eq. 13 is satisfied.

We reach the same result by studying Eq. 12. We can replace $P(\mathcal{R}(b) = \neg\text{kept})$ with $1 - P(\mathcal{R}(b_1) = \text{kept})$, which leads to the inequality:

$$\frac{e^{-k(\epsilon)\cdot(x-h_0)}}{1+e^{-k(\epsilon)\cdot(x-h_0)}} \leq \frac{e^{\epsilon - k(\epsilon)\cdot(y-h_0)}}{1+e^{-k(\epsilon)\cdot(y-h_0)}} + \delta \quad (19)$$

Repeating the analysis we did for Eq. 11, we show that the logistic functions never intersect when $y = x - 1$, while they may intersect when $y = x + 1$. We define $d_r(x)$ as $\frac{e^{-k(\epsilon)\cdot(x-h_0)}}{1+e^{-k(\epsilon)\cdot(x-h_0)}} - \frac{e^{\epsilon - k(\epsilon)\cdot(x+1-h_0)}}{1+e^{-k(\epsilon)\cdot(x+1-h_0)}}$. By solving $\frac{\partial d_r}{\partial x} = 0$, we obtain the solution $\bar{x_r} = \frac{1}{k}\cdot\ln(\frac{-r_r - \sqrt{r_r^2 - 4\cdot s_r\cdot q_r}}{2\cdot s_r})$, where $s_r = e^{k(\epsilon)\cdot h_0}\cdot(e^{\epsilon - k(\epsilon)} - 1)$, $r_r = k(\epsilon)\cdot e^{2\cdot k(\epsilon)\cdot h_0 - k(\epsilon)}\cdot(e^{\epsilon} - 1)$ and $q_r = e^{3\cdot k(\epsilon)\cdot h_0 - k(\epsilon)}\cdot(e^{\epsilon} - e^{-k(\epsilon)})$. In this case, r_r and q_r are positive while s_r is negative.

Finally, we observe that $d(\bar{x}) = d_r(\bar{x_r})$. Since d and d_r are symmetric over the axis passing in h_0, the two functions have the same maximum value. Moreover, $\bar{x}$ and $\bar{x_r}$ are the solutions to $\frac{\partial d}{\partial x} = 0$ and $\frac{\partial d_r}{\partial x} = 0$, respectively. □

Theorem 1 shows that the bin removal mechanism is (ϵ, δ)-differentially private. Since there are several scenarios where data curators prefer to have δ equal to zero, in the following theorem we define a set of conditions which guarantee $(\epsilon, 0)$-differential privacy.

Theorem 2. *$\mathcal{R}$ is $(\epsilon,0)$-differentially private when $\epsilon \geq k(\epsilon)$*

Proof. We focus the analysis on Eq. 11. As explained in the proof of Theorem 1, the case where the two logistic functions may intersect is the one where b_2 is smaller than b_1, i.e. $y = x - 1$. By setting $\delta = 0$, we can rewrite Eq. 13 as:

$$1 + e^{k(\epsilon) - k(\epsilon)\cdot(x-h_0)} - e^{\epsilon} - e^{\epsilon - k(\epsilon)\cdot(x-h_0)} \leq 0. \quad (20)$$

Since $e^{\epsilon} \geq 1$, $1 - e^{\epsilon}$ is negative or null. Therefore, the solutions of

$$e^{k(\epsilon) - k(\epsilon)\cdot(x-h_0)} - e^{\epsilon - k(\epsilon)\cdot(x-h_0)} \leq 0 \quad (21)$$

supply a sufficient condition for $(\epsilon, 0)$-differential privacy. Eq. 21 holds when $k(\epsilon) \leq \epsilon$. We reach the same conclusion by studying the other possible outcome of $\mathcal{R}$, i.e. Eq. 12. The analysis of Eq. 19 develops as the one of Eq. 13. □

One of the main results of Theorem 2 is that when ϵ is small and $\delta = 0$, $k(\epsilon)$ is small as well, i.e. the logistic function controlling the removal probability is slight. It is also worth noting that there are values of $k(\epsilon) > \epsilon$ which lead to $(\epsilon, 0)$-differential privacy, since Eq. 21 is stricter than Eq. 20.

Finally, we set a few conditions to determine the value of h_0. Ideally, we want h_0 as small as possible, to reduce the number of removed bins. At the same time, we want that $P(\mathcal{R}(b) = \text{kept})$ is close to 0 when the height of the bin b is close to 0. It follows that h_0 can be calculated by finding the minimum value of h_0 such that $1/(1 + e^{k(\epsilon)\cdot h_0}) \leq z$, where z is the probability that $\mathcal{R}$ does not remove an empty bin. It is worth noting that when $z = 0$, $h_0 = \infty$, since 0 is one of the asymptotes of the logistic function we defined. From the above inequality, we can isolate h_0:

$$h_0 = \frac{1}{k(\epsilon)}\cdot\ln\left(\frac{1-z}{z}\right) \quad (22)$$

Algorithm 2 Pseudocode of BD_{br} for histogram queries

1: $\mathbf{d_i} \leftarrow Q(D_i)$
2: **for all** $(b,h) \in \mathbf{d_i}$ **do** Add (b,h) to $\mathbf{c_i}$ with probability $1/(1 + e^{-k(\epsilon/(3\cdot w))\cdot(h-h_0)})$
3: $d \leftarrow |\mathbf{c_i}|$; $dis \leftarrow \frac{1}{d}\sum_{(b,h)\in\mathbf{c_i}} |\mathbf{last}[b] - h|$; $\lambda_{i,1} \leftarrow (3\cdot w)/(\epsilon \cdot d)$
4: $dis \leftarrow dis + Lap(\lambda_{i,1})$
5: $\epsilon_{rm} \leftarrow \epsilon/3 - \sum_{k=i-w+1}^{i-1} \epsilon_{k,2}$
6: $\lambda_{i,2} = 2/\epsilon_{rm}$
7: **if** $dis > \lambda_{i,2}$ **then** $\epsilon_{i,2} \leftarrow \epsilon_{rm}$;
8: **for all** $(b,h) \in (\mathbf{c_i})$ **do** $\mathbf{last}[b] = \mathbf{o_i}[b] \leftarrow h + Lap(\lambda_{i,2})$
9: **else** $\epsilon_{i,2} \leftarrow 0$; $\mathbf{o_i} \leftarrow null$
10: **return** $\mathbf{o_i}$

4.2 BD_{br}: w-event privacy with bin removal

Theorem 1 shows that the bin removal approach is ϵ-differentially private. The next step is to develop an extension of the BD scheme for w-event privacy to handle bin removal.

Algorithm 2 shows BD_{br}, the w-event privacy mechanism for computing histograms with **b**in **r**emoval, at the i-th iteration. Initially, the query is evaluated and all the non-empty bins (b,h) (where b is the bin identifier, and h is the bin height) are stored in $\mathbf{d_i}$. Next, the algorithm computes a subset $\mathbf{c_i} \subseteq \mathbf{d_i}$ by applying the bin removal mechanism described above (Line 2). The d parameter now can vary among evaluations and is set as the size of $\mathbf{c_i}$. Since the output will contain all the bins in $\mathbf{c_i}$, d does not reveal any additional information to the data analyst about $\mathbf{d_i}$.

The algorithm uses a data structure **last** to store the last non-null entry numbers released for every bin observed in the past. This data structure is used to compute the mean absolute error dis (Line 3). If **last** does not contain any information about a bin b, $\mathbf{last}[b]$ is set to 0 (as in the first iterations of Algorithm 1, when there are only null outputs). The value of dis is used to determine if the current iteration should produce a non-null release (Line 8). A significant difference between BD and BD_{br} is how the privacy budget is distributed. BD uses half of the budget to add noise to dis and part of the other half to obfuscate $\mathbf{c_i}$. BD_{br} allocates budget also to the bin removal mechanism: it equally distributes a third of the budget to the three privacy mechanisms it embeds. As in BD, the budget for obfuscating the non-null answers can be moved across datasets.

4.3 BD_{br} in SihlQL

We can now use BD_{br} to build a physical operator to evaluate $\gamma_{?bin,COUNT(*)}(\Omega_h)$, where Ω_h is the set of mappings containing the data to compute histograms. Since BD_{br} implements one algebra operator, it follows that it can be used to also evaluate the algebra expressions in Eq. 3 as well as other sub-expressions which lead to a bag Ω_h compliant with C1, C2, and C3. BD_{br} relaxes conditions C4 and C5, allowing to output histograms with a dynamic number of bins. As a consequence, the OPTIONAL and GRAPH clauses in Lines 7 and 8 of Listing 2 are not mandatory any more and can be omitted. Hence, BD_{br} extends the set of privacy-preserving queries that can be expressed in *SihlQL*.

In the next section, we investigate how Algorithm 2)differs from Algorithm 1 in terms of utility (or accuracy, as controlled by ϵ and w) of query results. Note that we do not run a traditional empirical evaluation of our contributions, as the privacy guarantees of our new mechanism—the main contribution of this paper—is already established by proofs provided in this section.

5 EXPERIMENTAL ANALYSIS

The main goal of this section is to provide insights about the bin removal mechanism in practice. We first analyse the bin removal mechanism alone and how different parameters affect it, and then we study its impact when integrated into *SihlQL*.

5.1 Analysis of the bin removal parameters

Our first analysis focuses on the bin removal mechanism and on the effect that the parameters $k(\epsilon)$, ϵ, δ and z have on it. Figure 1a shows the value of h_0 for different values of z and $k(\epsilon)$. We observe that the higher the value of $k(\epsilon)$, the less z influences h_0: when $k(\epsilon) \geq 1$, h_0 is having small variations (i.e. the shift of the probability curve is smaller). Hence, the lower $k(\epsilon)$, which regulates the steepness of the logistic function (and consequently the probabilities used in the Bernoulli distribution), the higher the influence of z, which entails the probability of removing an empty bin), on h_0. Figure 1b exemplifies the effect of z when $k(\epsilon)$ is fixed: the probability curve shifts to the right as z decreases. Hence, when the probability of removing an empty bin (i.e., z) decreases, the bigger the size of a bin that has a 50% chance of removal.

Figure 1c shows the effect of $k(\epsilon)$ on the probability curve. As explained above, $k(\epsilon)$ affects the steepness of the curve. While for very small values of $k(\epsilon)$ (i.e. $k(\epsilon) < 0.1$) the growth of the curve is very smooth, when $k(\epsilon) \geq 0.1$ the curve starts to become steeper, resembling the step function. The fact that for relatively small values of $k(\epsilon)$ the curve is steep indicates that the logistic function is a good choice to represent the bin removal probability.

The heat-maps in Figures 1d, 1e and 1f show how delta varies for different values of $k(\epsilon)$ and ϵ. The green area identifies the values for which the bin removal mechanism is $(\epsilon, 0)$-differentially private. The dashed line is the function $k(\epsilon) = \epsilon$, which we determined as sufficient condition for $(\epsilon, 0)$-differential privacy in Theorem 2. Figure 1b shows that when $\epsilon \geq 1$, the line is a good approximation of the maximum value of $k(\epsilon)$ that ensure $\delta = 0$. The heat-map also shows that the value of δ increases quickly as the difference between $k(\epsilon)$ and ϵ increases.

Figure 1c and 1d focus on values of $\epsilon \leq 1$: they show that in this area, there are values of $k(\epsilon) > \epsilon$ where $\delta = 0$. This happens because the area of $(\epsilon, 0)$-differential privacy is defined by Eq. 20, while Theorem 2 is based on Eq. 21, which is a stricter approximation. Hence, for small values of ϵ, a user may want to find a maximum value of $k(\epsilon)$ such that $\delta = 0$ in order to increase the steepness of the logistic curve and gain some utility.

In conclusion, we observe that when histograms have large bin sizes (in the order of hundreds of entries) our mechanism is unlikely to remove many bins. When bins have small sizes, the mechanism is likely to remove those and for very small sizes some hand-tuning of $k(\epsilon)$ parameters might be helpful. Depending on the use case, user may also decide to increase the value of δ to gain utility.

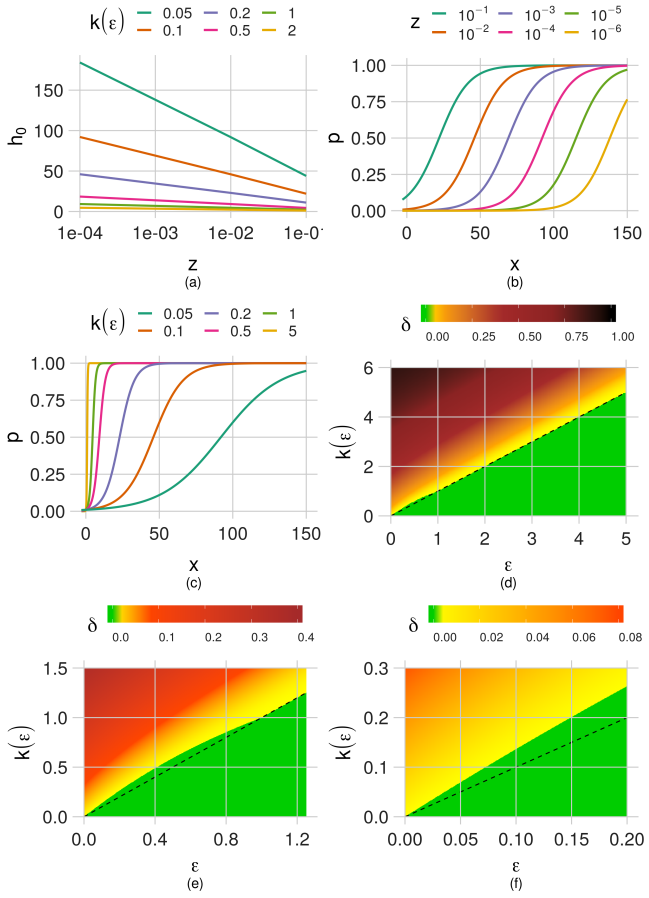

Figure 1: (a) effect of z on h_0; (b) effect of z on the bin removal probability ($k(\epsilon) = 0.1$); (c) effect of $k(\epsilon)$ on the bin removal curves; (d-f) values of δ for combinations of ϵ and $k(\epsilon)$.

5.2 Bin removal mechanism impact on SihlQL

In this section, we empirically illustrate the impact that the bin removal mechanism has on w-event privacy and *SihlQL*. We explained that the new mechanism increases the expressiveness of *SihlQL*. Intuitively, the price to pay for such expressiveness is a reduction of utility: we expect that results produced by BD are better than the ones produced by BD_{br}. The rationale for this expectation is that BD_{br} introduces a third privacy mechanism, requiring to split the available budget in three parts rather than two, leading to a higher amount of noise injected by each mechanism.

5.2.1 Data, queries and parameters. For our analysis, we consider two real-world datasets. The IPTV dataset contains a selection of viewership data from an IPTV platform. It covers a time period of about 12 hours and contains 49,035 distinct timestamps with more than 950 million events. The stream describes 450 channels, which are also stored in an RDF graph to be used as background data. We use two queries: to study BD, we use the query in Listing 2; to study BD_{br}, we run a query without channel graph and with the WHERE clause composed by one BGP to extract what users watch (the BGP inside the OPTIONAL clause in Line 7 of Listing 2). Hence, the queries build histograms with channels as bins and the number of current viewers as bin size.

The second dataset contains reviews data from Amazon [18]. We extracted a stream with 758,745 events in 1,266 stream items, which cover a period from 26 January 2002 to 23 July 2014. The stream contains 112 products. We run two queries analogous to the ones of the IPTV case, which count the number of reviews per product.

We consider four values for the budget parameter $\epsilon = \{0.1, 0.5, 1, 5\}$ and four values for the parameter $w = \{1, 5, 10, 100\}$. In settings where DP is applied to static datasets, ϵ in the order of tens is usually considered too high and assumed to lead to privacy leaks; values in single digit are usually adopted by industries (e.g. [25]), and values lower than 1 are associated to strong privacy. The presence of w makes the calibration of ϵ even harder since it directly affects the noise injected in the algorithms. In the BD_{br} experiments, we set $z = 10^{-2}$. We also set $k(\epsilon) = \epsilon$, i.e. $\delta = 0$, to focus on the cases where ϵ-indistinguishability must be guaranteed. Every experiment considers one dataset, one query, and a specific value of ϵ and w. We run every experiment twenty times.

5.2.2 The SihlQL engine. We developed a *SihlQL* engine prototype to run experiments[1]. The prototype is built on top of Apache Jena 3.6.0 and Apache Flink 1.7.1. Jena parses the query, creates the algebraic tree, and optimises it. The prototype replaces the algebraic operators with physical operators, implemented as Flink functions. The result of the compilation process is a topology, which can be submitted to Flink for execution. When the *SihlQL* query contains the PRIVACY clause, the prototype uses the physical operators implementing BD and BD_{br}; else it uses non-DP operators to compose the topology. Currently supported SPARQL operators include BGP, OPTIONAL, GRAPH, FILTER, GROUP BY, COUNT, and SUM.

We ran the experiments in a machine equipped with an Intel i7-6600 CPU (2.60 GHz, four logical CPUs) and 16GB RAM, which runs Ubuntu 19.04 with Oracle Java 1.8.0. Given that the main focus of our analysis is the trade-off between answer quality and expressiveness, we only report general time performance results. The SihlQL engine was able to process the IPTV dataset (i.e. 950 million events) in ca. 11 minutes, including the time to load the data (from a file stored in an external USB HDD storage) and store the results in an InfluxDB instance running in a server located in the institution local network. We noticed that the BD_{br} query was executed faster than the BD one (ca. 1 minute faster): it is not surprising since the former query processes only the items that appear in the stream recently, while the latter processes all the items at every new query evaluation. This suggests that the current prototype can process large amounts of data reactively, even if specific experiments are required to estimate accurate throughput and latency values (which is out of the scope of this study).

Both BD and BD_{br} queries were faster than the query without privacy, which was processed in ca. 14 minutes. This is due to the fact that the two privacy-preserving queries drops data during the execution, consequently increasing the time performance. In the next subsections, we focus on analysing the quality of the answers.

[1]The prototype and the Amazon dataset used in the experiments are available at: https://gitlab.ifi.uzh.ch/DDIS-Public/sihlql.

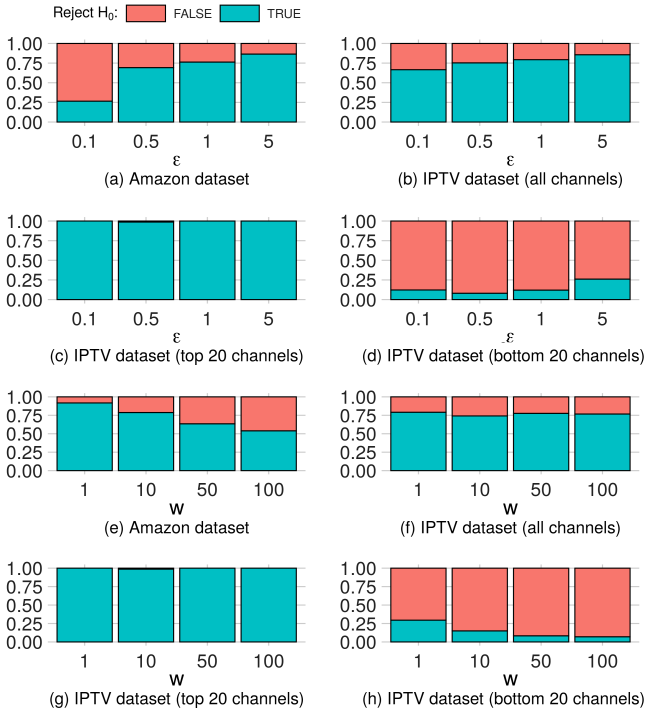

Figure 2: Percentage of items for which the privatised time series distances to the actual time series are indistinguishable for the Amazon dataset (a,e) and the IPTV dataset: all (b,f), top 20 (c,g) and the bottom 20 (d,h) TV channels.

5.2.3 Analysis design. Directly comparing BD and BD_{br} is not straightforward due to the fact that they execute different queries. Hence, they produce results at different time instants and on a different number of items. Therefore, we developed a comparison strategy based on the distance (or error) of the results from the original data as follows. For each item in the dataset, we computed the actual answer, which can be represented as a time series where values are the bin heights at different time points. Similarly, we can represent the outcomes of BD and BD_{br} as two collections of time series. We compare the two solutions by computing the distances between the privatised noisy time series and the actual time series.

Since both BD and BD_{br} include mechanisms to reduce the number of answers, the privatised time series have fewer time points than the actual one. We considered two methods to impute the missing points of the privatised time series: ARIMA [22] and last observation carried forward (LOCF). ARIMA is a standard time-series interpolation method. LOCF captures one of the basic ideas of w-event privacy by assuming that the current value is equivalent to the last non-null output. In our experiments, LOCF showed better results, i.e. closer distances, than ARIMA; therefore, in the following, we report the results obtained by imputing the missing points with LOCF. Since the final goal is to have a good approximation of the actual time series, we did not leverage sampling methods to reduce their points.

We used the Euclidean distance as the distance measure. We did not use other time series distances, such as DTW, because we know that the time series are aligned over the time dimension. As usually done in time series analysis, we applied z-normalisation before imputing the missing points and computing the distances. Running the experiments for each privacy approach (either BD or BD_{br}), item (product or channel, depending on dataset), ϵ, and w resulted in 20 distances. To compare BD and BD_{br}, we test if the sets of distances with the same item, ϵ, and w values are indistinguishable (the null hypothesis H_0) or not (the alternative hypothesis H_a). We employ the Mann-Whitney U test, a non-parametric test to assess if the distributions of two populations are equal or not.

5.2.4 Hypothesis testing results. Figure 2 shows the test results aggregated over ϵ and w. In each bar, the red area (at the top) shows the percentage of items for which we cannot reject the null hypothesis of the Mann-Whitney U test, i.e. the distances between BD and the original data are indistinguishable from the ones between BD_{br} and the original values. Figures 2a and 2b show the results for the Amazon and IPTV datasets aggregated over ϵ: the higher the privacy (i.e. the smaller ϵ), the more indistinguishable the results of the two privacy mechanisms. Figure 2e shows that w behaves similarly to ϵ in the Amazon dataset: when noise increases (i.e. w increases) the number of indistinguishable items increases as well. However, we cannot identify the same trend in Figure 2f, related to the IPTV dataset: w does not seem to affect the plotted ratio. This could be motivated by the more variability in the average and the maximum number of viewers per channel of the IPTV dataset.

To further investigate this, we analysed the behaviour of the 20 channels with the highest and the lowest median values of viewers, depicted in Figure 2(c,g) and 2(d,h) respectively. The plots show that for channels with a high number of viewers, BD and BD_{br} are producing results with different distances to the actual values, while for channels with a few viewers, it is the opposite. The same analysis on the Amazon dataset shows similar results, even if the ratios are less extreme than the IPTV case. This is because the number of products, as well as the median of their reviews for products, are smaller than the number of channels and their viewers.

This analysis partially confirms our initial intuition: when the height of the items (i.e. the size of histogram bin) is high, or the privacy-constraints are weak, BD and BD_{br} produce different results. However, when the height of the bins is small, and the privacy requirements strong, the two solutions produce similar results. Hence, when we put high privacy requirements (i.e. $\epsilon \leq 1$) on the publication of a data stream the utility of BD_{br} is comparable to BD's in 75% (respectively, 35% or 30%) of the cases—a positive results given that BD_{br} provides a higher expressiveness. The same is sometimes true to a lesser degree with large window sizes.

We observed than when $\epsilon = 0.1$, the BD execution for the Amazon dataset did not produce any non-null answer for $w \in \{1, 10\}$. The reason is that most of the items of this dataset have low values, and the condition at Line 7 of Algorithm 1 is never satisfied. BD_{br} does not suffer the same problem since the computation of *dis* considers only the items that have a height greater than 0 and that are not removed by the bin removal mechanism. This suggests that when the data has multiple bins with a low number of items, BD_{br} may be a more suitable solution than BD.

5.2.5 Difference in distances. When BD and BD_{br} are statistically indistinguishable, then BD_{br} higher expressiveness shows a clear

advantage. This sub-section further investigates what happens, when the two mechanisms are distinguishable, to understand further trade-off when using the methods. Figure 3 graphs the difference between BD_{br} and BD distances to the actual data for the IPTV dataset (for brevity, we omit the plot of the Amazon dataset, which is very similar). For each (ϵ, w) pair, we build the plot using the items that show distinguishable distances (i.e., rejected H_0 in the Mann-Whitney U test). Positive differences indicate that BD outputs are closer to the actual time series than BD_{br}; negative differences indicate the opposite case. We use relative distances, i.e. each distance is divided by the values of the actual time series[2].

The figure shows that the differences between the distances vary between -1 and 1. Even if in some cases the differences are large (close to 1, i.e., we can observe a difference up to the size of the bin), the figure shows that bins with large sizes, coloured blue, are in most of the cases close to the zero (i.e. similar error), while the ones with small size shift more. One reason for this behaviour is that the amount of noise added to a bin depends on the privacy budget but not on the actual bin size. Hence, smaller bins are more affected.

It is interesting to observe that for small values of ϵ and w, there are several cases where BD_{br} outperforms BD. This is positive since ϵ should be tuned to enforce strong privacy. When w increases, the differences shift towards positive values, and BD outperforms BD_{br}: when w increases, the BD_{br} bin removal mechanism has less budget and consequently filters out more items. This is especially evident for channels with fewer than 100 viewers (i.e., items with bins sizes $h \leq 100$; white and red in the figure), which suggests a decrease of the utility of BD_{br} related to w for small bin sizes. BD also improves over BD_{br} also when ϵ increases. When privacy is very weak ($\epsilon = 5$), almost all the distances are positive. This is because the BD mechanism that creates the noisy answers usually has more budget available than the BD_{br}'s, as BD_{br} constantly invests a third of the budget in the removal mechanism.

To summarise, BD_{br} includes an additional privacy mechanism. In addition to the advantages in terms of query expressiveness, the analyses show that the way in which those three mechanisms interact leads to benefits (sometimes even higher utility than BD), despite of the less available budget each of them has. As future research, we plan to further investigate these interactions and develop more sophisticated budget assignment policies, which may improve the performance of the solution.

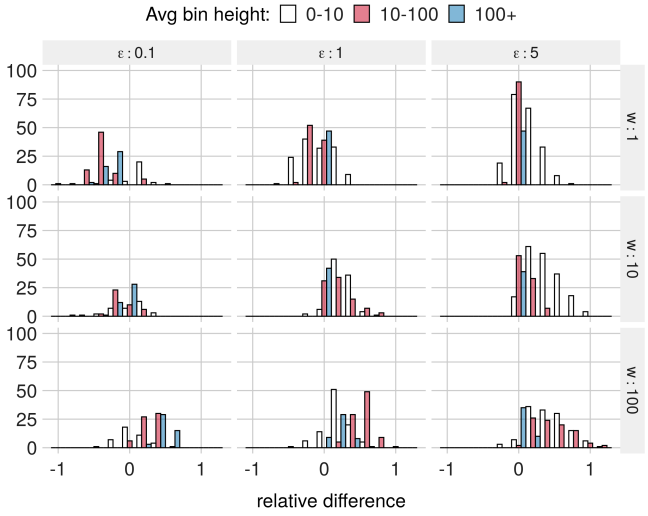

Figure 3: Difference of distances for distinguishable results.

6 CONCLUSIONS

The ability to exchange data and information is one of the pillars of our digital society. Data analyses lead to the creation of new knowledge, which in turn leads to innovation and ultimately to increased welfare. This also holds for the studies involving sensitive data, which are the key to explain to unlock the potential in a multitude of domains ranging from clinical studies, behavioural analyses, However, such analyses should be built keeping privacy as a key priority—indeed some argue as a basic human right—to avoid that sensitive information or secrets get leaked and misused.

In this article, we presented a semantic-web-based framework that allows experts to analyse and publish sensitive data Specifically, we introduced *SihlQL*, which allows defining queries over data streams such that answers have formal statistical guarantees against privacy leaks. *SihlQL* builds on top of SPARQL and the w-event differential privacy framework. To generalise *SihlQL* and improve the set of queries that can be privatised, we extended w-event privacy with a bin removal mechanism. We prove that our extension BD_{br} satisfies differential privacy constraints and, hence, provides statistical privacy guarantees. We also developed a prototype for *SihlQL* and showed that it can process streams with millions of events while privatising the query results.

Obviously, *SihlQL* can be improved. We aim to extend *SihlQL* with stream-specific operators, e.g. sliding windows and event pattern matching. This requires studying those operators' privacy implications, potentially rethinking how they work and how they can leak information. Currently, *SihlQL* builds on two of the foundational semantic web blocks—RDF and SPARQL—and we believe that other technologies can play an important role. Ontology- and constraint-languages, such as OWL and SHACL, would be ideal to describe the privacy aspects related to data and schema, automating privacy assessments over queries before and during their execution. These explorations will help to make *SihlQL* more versatile.

As such, the presentation of *SihlQL* and its expressiveness enhancing BD_{br} mechanism is a step towards the exploration of privacy-preserving techniques in the Web of Data. Given that semantic web research has so far mostly focused on publishing privacy-insensitive data, this is an important step for the field that will hopefully pave the way to a privacy-preserving web of data.

Acknowledgements. We thank the Swiss National Science Foundation for the partial support under contract number #407550_167177, and Genistat AG for their help with the IPTV dataset.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *ACM Conference on Computer and Communications Security*. ACM, 308–318.

[2] As a reference: for our datasets, the mechanisms' outputs relative distance to the actual time series varies between ca. 0 and 2 (for different privacy parameter combinations).

[2] Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. 2011. EP-SPARQL: a unified language for event processing and stream reasoning. In *WWW*. ACM, 635–644.

[3] Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. *VLDB J.* 15, 2 (2006), 121–142.

[4] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. 2010. Querying RDF streams with C-SPARQL. *SIGMOD Record* 39, 1 (2010), 20–26.

[5] Davide Francesco Barbieri and Emanuele Della Valle. 2010. A Proposal for Publishing Data Streams as Linked Data - A Position Paper. In *LDOW*, Vol. 628. CEUR-WS.org.

[6] Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. 2013. Private decayed predicate sums on streams. In *ICDT*. ACM, 284–295.

[7] Daniele Dell'Aglio, Emanuele Della Valle, Jean-Paul Calbimonte, and Oscar Corcho. 2014. RSP-QL Semantics: A Unifying Query Model to Explain Heterogeneity of RDF Stream Processing Systems. *Int. J. Semantic Web Inf. Syst.* 10, 4 (2014), 17–44.

[8] Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. 2017. Stream reasoning: A survey and outlook. *Data Sci.* 1, 1-2 (2017), 59–83.

[9] C. Dwork, M. Naor, T. Pitassi, and G.N. Rothblum. 2010. Differential privacy under continual observation. In *STOC*. 715–724.

[10] Cynthia Dwork and Aaron Roth. 2014. *The Algorithmic Foundations of Differential Privacy*.

[11] Giulia C. Fanti, Vasyl Pihur, and Ulfar Erlingsson. 2016. Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries. *PoPETs* 2016, 3 (2016), 41–61.

[12] Javier D. Fernández, Axel Polleres, and Jürgen Umbrich. 2015. Towards Efficient Archiving of Dynamic Linked Open Data. In *DIACRON@ESWC*. CEUR-WS.org, 34–49.

[13] Steve Harris and Andy Seaborne. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C. https://www.w3.org/TR/sparql11-query/

[14] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially Private Event Sequences over Infinite Streams. *PVLDB* 7, 12 (2014), 1155–1166.

[15] Sabrina Kirrane, Serena Villata, and Mathieu d'Aquin. 2018. Privacy, security and policies: A review of problems and solutions with semantic web technologies. *Semantic Web* 9, 2 (2018), 153–161.

[16] Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. 2011. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In *ISWC (1)*, Vol. 7031. Springer, 370–388.

[17] Alessandro Margara, Gianpaolo Cugola, Dario Collavini, and Daniele Dell'Aglio. 2018. Efficient Temporal Reasoning on Streams of Events with DOTR. In *ESWC*. Springer, 384–399.

[18] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. ACM, 165–172.

[19] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD Conference*. ACM, 19–30.

[20] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy*. IEEE, 111–125.

[21] Nigel Shadbolt, Kieron O'Hara, Tim Berners-Lee, Nicholas Gibbins, Hugh Glaser, Wendy Hall, and m c schraefel. 2012. Linked Open Government Data: Lessons from Data.gov.uk. *IEEE Intelligent Systems* 27, 3 (2012), 16–24.

[22] Robert H. Shumway and David S. Stoffer. 2005. *Time Series Analysis and Its Applications*. Springer-Verlag, Berlin, Heidelberg.

[23] Roney Silva, Bruno Leal, Felipe Brito, Vânia Maria Vidal, and Javam C. Machado. 2017. A Differentially Private Approach for Querying RDF Data of Social Networks. In *IDEAS*. 74–81.

[24] Latanya Sweeney. 1997. Weaving technology and policy together to maintain confidentiality. *J. of Law, Medicine and Ethics* 25, 2–3 (1997), 98–110.

[25] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. 2017. Privacy Loss in Apple's Implementation of Differential Privacy on MacOS 10.12. *CoRR* abs/1709.02753 (2017).

[26] Yuting Zhao, Guido Vetere, Jeff Z. Pan, Alessandro Faraotti, Marco Monti, and Honghan Wu. 2015. Meta-Level Properties for Reasoning on Dynamic Data. In *JIST*. Springer, 271–279.