# Precision-Bounded Access Control Using Sliding-Window Query Views for Privacy-Preserving Data Streams

Zahid Pervaiz, Arif Ghafoor, *Fellow, IEEE*, and Walid G. Aref, *Senior Member, IEEE*

**Abstract**—Access control mechanisms and Privacy Protection Mechanisms (PPM) have been proposed for data streams. The access control for a data stream allows roles access to tuples satisfying an authorized predicate sliding-window query. Sharing the sensitive stream data without PPM can compromise the privacy. The PPM meets privacy requirements, e.g., $k$-anonymity or $l$-diversity by generalization of stream data. Imprecision introduced by generalization can be reduced by delaying the publishing of stream data. However, the delay in sharing the stream tuples to achieve better accuracy can lead to false-negatives if the tuples are held by PPM while the query predicate is evaluated. Administrator of an access control policy defines the imprecision bound for each query. The challenge for PPM is to optimize the delay in publishing of stream data so that the imprecision bound for the maximum number of queries is satisfied. We formulate the precision-bounded access control for privacy-preserving data streams problem, present the hardness results, provide an anonymization algorithm, and conduct experimental evaluation of the proposed algorithm. Experiments demonstrate that the proposed heuristic provides better precision for a given data stream access control policy as compared to the minimum or maximum delay heuristics proposed in existing literature.

**Index Terms**—Privacy, *k*-anonymity, access control, data stream

## 1 Introduction

Data Stream Management Systems (DSMS) have been proposed to process transactional data, e.g., internet traffic, health monitoring, and sensor networks [1], [2]. Access control mechanisms for data streams ensure that only the authorized parts of the stream are available to each user or role [3], [4]. Objects protected under the access control mechanism are the queries or views of the data stream [4]. If the sensitive information in the authorized view of the data stream is not privacy protected, then the privacy of a person can be compromised even in the presence of access control. For example, for health monitoring and epidemic surveillance applications, various stakeholders including public health officials, epidemiologists, doctors, and researchers from various agencies and regions, should only access the views of patient streaming data for which they have the authorization. Furthermore, prior to publishing these views to the authorized stakeholders, privacy of patients must be protected. The well-known privacy preservation techniques of $k$-anonymity [5] and $l$-diversity [6] have also been used for privacy protection of data streams [7], [8]. However, to the best of our knowledge, precision-bounded access control along with privacy protection has not been investigated before for data streams. The focus of this paper is to develop a privacy-preserving mechanism for publishing precision-bounded authorized views of streaming data. The existing approaches for privacy protection of data streams suppress the time-stamp attribute [7], [8]. However, applications like access control do require time-stamp information to evaluate temporal queries over stream data.

The attribute values in data stream tuples can be generalized to satisfy given privacy requirements. Generalization of relational data attributes introduces imprecision in the query results for access control mechanism. This imprecision can be reduced if the publishing of stream data is delayed. However, the delay introduces False-Negatives (FNs) in the query results in case the tuples satisfying the query predicate are not made available to the access control mechanism at the instance of query evaluation.

In the following example (Example 1), we provide a motivating scenario and explain how imprecision in an authorized view can be used to assess the utility of the streaming data. Subsequently, in Example 2, we illustrate how a privacy-preserving mechanism can be applied prior to generating such authorized views. Through these examples we highlight the requirement for defining the imprecision bounds for publishing authorized data stream views.

**Example 1 (Motivating Scenario).** Syndromic surveillance systems have been developed by state and federal agencies to detect and monitor public health emergencies [9]. The emergency department data (age, gender, location, time of arrival, symptoms, etc.) from county hospitals is collected and is sent to the department of health at the state level on an hourly basis. The surveillance data stream is classified into syndrome categories and is

• *Z. Pervaiz and A. Ghafoor are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907. E-mail: {zpervaiz, ghafoor}@purdue.edu.*
• *W. Aref is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907. E-mail: aref@cs.purdue.edu.*

*Manuscript received 17 Oct. 2014; accepted 17 Dec. 2014. Date of publication 12 Jan. 2015; date of current version 1 June 2015.*
*Recommended for acceptance by J. Pei.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TKDE.2015.2391098*

| Role | Designation |
|---|---|
| SE | State Epidemiologist |
| CE1 | County 1 Epidemiologist |
| CE2 | County 2 Epidemiologist |

| TC | Type | Definition |
|---|---|---|
| T1 | Sliding Window | Size = 24 hours, Step = 4 hours |

| Permission | Authorized Query Predicate (View) |
|---|---|
| P1 | Location = County 1 ∧ Age = 15-65 ∧ Syndrome = Influenza ∧ T1 |
| P2 | Location = County 2 ∧ Age = 15-65 ∧ Syndrome = Influenza ∧ T1 |

$$\begin{tikzcd} & \text{SE} \arrow[dl] \arrow[dr] & \\ \text{CE1} \arrow[d] & & \text{CE2} \arrow[d] \\ \text{P1} & & \text{P2} \end{tikzcd}$$

Fig. 1. Access control policy.

anonymized by the state department of health. The data is subsequently shared with the department of health in each county.

A Role Based Access Control (RBAC) policy is given in Fig. 1. Role SE is above roles CE1 and CE2 in the role hierarchy and can execute all the permissions allowed to roles CE1 and CE2. This policy allows the users to access the data stream view defined by the authorized queries, e.g., role CE1 can view tuples under Permission P1 over the data stream in a 24-hour window with a slide of 4 hours (i.e., updated every four hours). The Temporal Constraint (TC) T1 defines a sliding-window (size $=$ 24 hours, slide $=$ 4 hours) of stream data upon which the query can execute. Permissions under an access control policy ensure that only the authorized view of the data stream is available to each user. Anonymization adds False-Positives (FPs) to the authorized view and the precision can be improved by delaying the stream data. However, the delay adds false-negatives for the views (when a tuple satisfying the view is not shared). The imprecision bound for each permission ensures that the authorized view is within the required tolerance at the time of predicate evaluation. The total imprecision for a view is the sum of false-positives and false-negatives and is used to gauge the utility of the authorized view.

The contributions of the paper are as follows. First, we introduce the concept of precision-bounded access control for privacy-preserving data streams. Second, we formulate the Precision-bounded Access Control for privacy-preserving data strEams (PACE) problem and give hardness results along with probabilistic analysis for query bound violation. Third, we propose a heuristic for an approximate solution of the PACE problem and conduct empirical evaluation.

The rest of this paper proceeds as follows. In Section 2, relevant background is discussed. The problem formulation and definitions are presented in Section 3. Section 4 covers the proposed Total Imprecision Minimization (TIM) heuristic for multi-dimensional partitioning of stream data to satisfy imprecision bounds for predicate sliding-window queries. Experimental results are presented in Section 5. The related work is presented in Section 6 and Section 7 concludes the paper.

## 2 Background

Given a stream $T[i] = \{ID, TS, A_1, A_2, \ldots, A_d, SA\}$, where ID is an identity attribute, TS is a time-stamp attribute that represents the arrival time of a tuple, $A_j$ is a Quasi Identifier (QI) attribute, SA is the sensitive attribute, $d$ is the number of QI attributes, and, $i$ is the current time-stamp. $T[i]$ represents all the data stream tuples that have arrived till the time instance $i$. The identity attribute (e.g., social security number) can uniquely identify an individual in a data stream. QI attributes (e.g., address, age) can be used with the background information to identify an individual even if the identity attribute has been suppressed [5]. If the sensitive attribute SA is associated with a unique individual, it results in privacy violation.

### 2.1 Privacy Definitions

$k$-anonymity [5] for streaming data has been proposed by Cao et al. [7] and Zhou et al. [8]. Both have suggested to suppress the TS attribute in the anonymized stream for privacy protection. However, with respect to access control over streaming data, the TS attribute is required for the evaluation of sliding-window queries. We propose that the generalized time-stamp value for each Equivalence Class (EC) must be included in the anonymized stream. The time-stamp attribute is a quasi-identifier attribute as knowing the time-stamp value for a person in a relational stream data can allow to find the associated sensitive value and can result in violation of privacy of that person.

**Definition 1 (Equivalence Class).** *An equivalence class is a set of tuples having the same QI attribute and time-stamp value.*

**Definition 2 (Stream $k_s$-anonymity Property [8]).** *A data stream $T^p[i]$ satisfies the $k_s$-anonymity property if each published equivalence class has $k$ or more tuples and if $t_1.ID = t_2.ID$ then $EC(t_1) \neq EC(t_2)$ for any $t_1, t_2 \in T[i]$.*

Here, $T^p[i]$ is the anonymized view of the stream data that is published till time instance $i$. In $T^p[i]$, the identity attribute is suppressed, the QI and the time-stamp attribute values are generalized and the sensitive attribute is published. The time-stamp attribute gives the arrival time of a tuple $t \in T[i]$. The delay in publishing is equal to $t$.PUB - $t$.TS, where $t$.PUB is the time when a tuple is published (i.e., is added to $T^p[i]$). The second part of the stream $k_s$-anonymity definition provides the constraint that two tuples with same ID must be in different equivalence classes. In the case of a data stream, multiple tuples ($\geq k_s$) can be received from the same person/ID in a short span of time. Without the above constraint, the tuples with the same ID can be put into the same equivalence class without any generalization resulting in a high probability of privacy violation. $l$-diversity counters the homogeneity attacks possible against $k$-anonymity when all the tuples in an equivalence class have the same sensitive attribute value.

**Definition 3 (Stream $l_s$-Diversity Property).** *A data stream $T^p[i]$ satisfies the $l_s$-diversity property if each equivalence class has at least $l$ distinct values of the sensitive attribute and if $t_1.ID = t_2.ID$ then $EC(t_1) \neq EC(t_2)$ for any $t_1, t_2 \in T[i]$.*

In the case of sensitive numeric attributes, if the numeric values in an equivalence class are close to each other, an $l_s$-diverse equivalence class can still leak private information. Variance diversity [10] and $t$-closeness [11] have been proposed to protect privacy against such a threat. We define the stream variance diversity as follows:

**Definition 4 (Stream Variance Diversity).** *A data stream $T^p[i]$ is variance diverse if the variance $V_s(EC)$ of each published equivalence class satisfies $V_s(EC) \geq v_s$, where $v_s$ is the data stream variance diversity parameter. In addition, if $t_1.ID = t_2.ID$ then $EC(t_1) \neq EC(t_2)$ for any $t_1, t_2 \in T[i]$.*

**Definition 5 (Delay Constraint).** *The delay constraint, denoted by δ, is the maximum delay before which a tuple is required to be published.*

The delay constraint can be set according to the data stream application requirement regarding availability of the anonymized tuples. The time constraint set by $\delta$ ensures that the delayed tuples eventually get published. $\delta$ can also be set based on storage limitations of the system anonymizing sensitive stream data. $T^h[i]$ is the set of tuples from $T[i]$ that are put on hold and are still to be anonymized at time instance $i$.

**Example 2 (Privacy-preserving Data Stream Views).** In this example we consider a sensitive data stream and provide authorization views for roles defined in Example 1. Fig. 2a lists the data stream tuples $T[4]$ received till time instance 4, and Fig. 2b gives the corresponding anonymized data stream $T^p[4]$. The data stream in Fig. 2a does not satisfy $k$-anonymity because knowing the age and zip code of a person allows associating a disease to that person. The data stream $T^p[4]$ in Fig. 2b is two-anonymous and two-diverse. The tuples $T^h[4]$ in Fig. 2c are on hold and are still waiting to be published. Assuming time stamps 1-4 are inside the active sliding window, County 1 zip is 15, and County 2 zip is 28 then the privacy-preserved data stream views for role CE1, CE2 and SE (refer Fig. 1) at time stamp 4 are given in Fig. 2d. It can be noted that imprecision has been added after anonymization to the authorized views for roles. The main challenge addressed in this paper is that imprecision added to each anonymized view should be less than the imprecision bound specified for that view.

| | | $QI_1$ | $QI_2$ | SA |
|---|---|---|---|---|
| ID | TS | Age | Zip | Disease |
| A | 1 | 5 | 15 | Flu |
| B | 1 | 15 | 25 | Fever |
| C | 2 | 28 | 28 | Diarrhea |
| D | 2 | 25 | 15 | Fever |
| E | 3 | 22 | 28 | Flu |
| F | 3 | 32 | 35 | Fever |
| G | 4 | 38 | 32 | Flu |
| H | 4 | 35 | 25 | Diarrhea |
| ... | ... | ... | ... | ... |

(a) Sensitive data stream $T[4]$

| | | $QI_1$ | $QI_2$ | SA | |
|---|---|---|---|---|---|
| ID | TS | Age | Zip | Disease | |
| A | 1-2 | 5-25 | 15-15 | Flu | |
| D | 1-2 | 5-25 | 15-15 | Fever | TS = 2 |
| C | 2-3 | 22-28 | 28-28 | Diarrhea | |
| E | 2-3 | 22-28 | 28-28 | Flu | TS = 3 |
| F | 3-4 | 30-40 | 20-40 | Fever | |
| G | 3-4 | 30-40 | 20-40 | Flu | TS = 4 |
| ... | ... | ... | ... | ... | |

(b) 2-anonymous data stream $T^p[4]$

| | | $QI_1$ | $QI_2$ | SA |
|---|---|---|---|---|
| ID | TS | Age | Zip | Disease |
| B | 1 | 15 | 25 | Fever |
| H | 4 | 35 | 25 | Diarrhea |

(c) Stream data on hold $T^h[4]$

| Roles | TS | Age | Zip | Disease |
|---|---|---|---|---|
| CE1 → | 1-2 | 5-25 | 15-15 | Flu |
| CE2 → | 2-3 | 22-28 | 28-28 | Flu |
| SE → | 1-2 | 5-25 | 15-15 | Flu |
| | 2-3 | 22-28 | 28-28 | Flu |

(d) Data stream view for roles at TS = 4

Fig. 2. Generalization of streaming data for $k$-anonymity and $l$-diversity.

## 2.2 Stream Query Model

Predicate window queries have been proposed for streaming data management systems [12]. Other types of queries on streaming data are the snapshot query [13] and the landmark query [14]. The sliding window query is defined by two parameters: 1) *Range* that defines the size of query window and 2) *Slide* that defines the step by which the window moves [12], [14]. If the slide of the window is less than the range, then the query sliding windows overlap. Otherwise, if the slide is equal to the range, then the windows are non-overlapping and are also known as tumbling windows. The sliding-window query can be either a tuple-count sliding-window or a time-sliding-window [1]. In this paper, we consider time-sliding-window queries. The predicate sliding-window query is evaluated at the end of the window size and then the window slides by the step size.

## 2.3 Role Based Access Control

**Definition 6 (RBAC Policy).** *An RBAC policy ρ is a tuple $\langle U, R, P, UA, PA, RH \rangle$, where U is a set of users, R is a set of Roles, P is a set of Permissions, RH is a Role Hierarchy that is a partial order on roles, UA is a user-to-role assignment relation, and RA is a role-to-permission assignment relation [15].*

Role-based Access Control for data streams has been proposed by Carminati et al. [4]. Permissions under P are the sliding-window query predicates that define the authorized view of the data stream.

## 3 Precision-Bounded Access Control for Privacy-Preserving Data Streams

In Section 3.1 we discuss query evaluation semantics and give definitions for the imprecision, imprecision bound and average query bound violation (AQV). We discuss precision-bounded access control enforcement and the choice of relevant semantics in Section 3.2. In Section 3.3 we formulate the PACE problem and present its hardness results. Section 3.4 presents the probabilistic analysis for query bound violation for a given imprecision bound. An access control policy administrator can use this analysis to revise the imprecision bounds for the queries if the probability of satisfying the bound for a large number of queries at any time instance is very low.

## 3.1 Predicate Evaluation and Imprecision

A predicate sliding-window query is evaluated for a data stream, say $T[i]$, by including all the stream tuples that satisfy the query predicate. For predicate evaluation over an anonymized data stream $T^p[i]$, we adhere to the same semantics as suggested in [10], [16], i.e., include all the tuples in equivalence classes that overlap the query predicate range. We refer to query evaluation under these semantics as the *Overlap* semantics. Another possible semantics for query evaluation can be to include all tuples in the equivalence classes that are fully enclosed inside the query predicate range. These semantics are referred to as the *Enclosed* semantics. The definitions in the following paragraphs and the anonymization algorithm in Section 4 follow the *Overlap* semantics. We compare the two semantics in Section 3.2 for access control enforcement.

**Definition 7 (False-Positive Tuple).** *A tuple is a false-positive when it does not satisfy the sliding-window query predicate at the time instance of query evaluation but is included in the query result as the equivalence class in $T^p[i]$ that contains the tuple overlaps the query predicate.*

The number of False-Positive tuples in the result of a predicate sliding-window query, say $Q_j[i]$, at any time instance $i$, is as follows:

$$FP_{Q_j[i]} = |Q_j(T^p[i])| - |Q_j(T[i] - T^h[i])|, \text{where} \tag{1}$$

$$|Q_j(T^p[i])| = \sum_{EC(\text{overlaps})\ Q_j} |EC|$$

A published partition can add a false-positive tuple to a predicate sliding-window query due to a spatial overlap (QI attributes), temporal (time-stamp attribute) overlap, or both temporal and spatial overlaps.

**Definition 8 (False-Negative Tuple).** *A tuple is a false-negative when it satisfies the predicate sliding-window query at the time instance of query evaluation but is not included in the query result due to being put on hold.*

The number of False-Negative tuples for a query, say $Q_j[i]$, evaluated at time instance $i$, is as follows:

$$FN_{Q_j[i]} = |Q_j(T^h[i])|. \tag{2}$$

If the publishing delay is increased, the number of false-positives reduces because equivalence classes with less imprecision can be formed while at the same time the number of false-negatives increases.

**Definition 9 (Sliding-Window Query Imprecision).** *Query imprecision is defined as the total sum of false-positives and false-negatives for a predicate sliding-window query evaluated on an anonymized stream $T^p[i]$ at any given time instance $i$.*

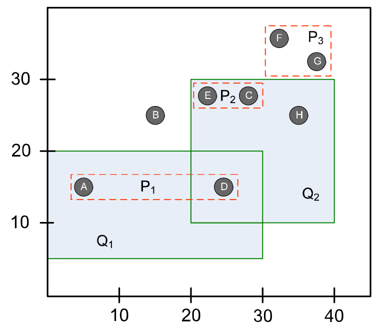


Fig. 3. Query evaluation over an anonymized data stream.

*The imprecision for query $Q_j[i]$ evaluated at time instance $i$ is denoted by $imp_{Q_j[i]}$ and is equal to the sum of false-positives and false-negatives. In other words,*

$$imp_{Q_j[i]} = FP_{Q_j}[i] + FN_{Q_j}[i]. \tag{3}$$

Here, query $Q_j[i]$ is evaluated over $T^p[i]$ by including all the tuples in equivalence classes that overlap the query region and $T^h[i]$. $imp_{Q_j[i]}$ is a utility measure that for a given sliding-window query captures two types of information loss; loss due to generalization (in terms of false positives) and loss due to publishing delay (in terms of false negatives).

**Example 3.** The two-dimensional representation for the quasi-identifier attributes of the anonymized data stream in Fig. 2 is given in Fig. 3. The rectangles with solid lines represent queries $Q_1$ and $Q_2$. The rectangles with dotted lines represent partitions (equivalence class). Assume that Query $Q_2[4]$ is evaluated at time instance 4. There is one false-positive Tuple $A$ for $Q_2[4]$ as Partition $P_1$ and $P_2$ overlap the query. The false-negative tuple for $Q_2[4]$ is Tuple $H$ as that tuple is still to be published and has not joined any equivalence class.

**Definition 10 (Query Imprecision Bound).** *The query imprecision bound, denoted by $B_{Q_j[i]}$, is the total imprecision acceptable to the access control mechanism when the sliding-window query predicate $Q_j[i]$ is evaluated at time instance $i$.*

A query violates the imprecision bound when, at the time of query evaluation, the total imprecision is more than the imprecision bound. For a sliding-window query, the query evaluation takes place at each step. Accordingly, we define the average query bound violation as follows:

**Definition 11 (Average Query-bound Violation (AQV)).** *The Average Query-bound Violation for a query $Q_j$ is the average number of times the query imprecision bound is violated over a given time period. In other words,*

$$AQV_{Q_j} = \frac{V_{Q_j}}{N_{Q_j}}, \tag{4}$$

*where $N_{Q_j}$ is the number of steps Query $Q_j$ takes till the current time instance and $V_{Q_j}$ is the number of times the imprecision bound is violated for these steps.*

AQV treats violation of the imprecision bound by a query at each step equally even if the amount of violation differs dramatically. However, the focus in this paper is on meeting

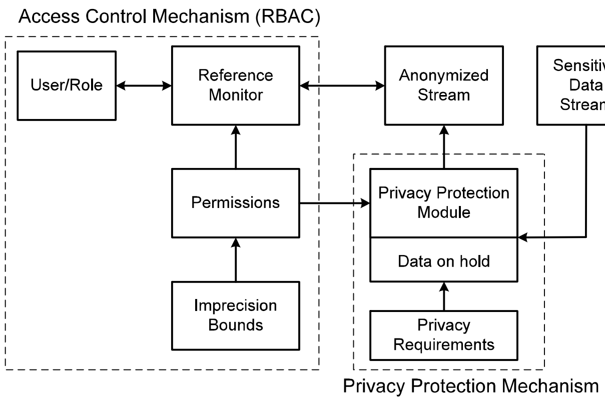


Fig. 4. Precision-bounded access control enforcement.

the precision bounds and AQV precisely captures the average performance of queries in terms of bound violations.

**Example 4.** Consider a sliding-window query, say $Q_1$, that takes 10 steps during a given time interval. At each query step, the query imprecision is evaluated and the imprecision value violates the imprecision bound at 4 of these steps. The average query bound violation for $Q_1$ is then 0.4 (4/10). Consider another query, say $Q_2$, that takes 20 steps during the same time interval and the imprecision bound is violated at 6 of these steps. The average query bound violation for $Q_2$ is then 0.3 (6/20) and, on average, $Q_2$ has better accuracy than that of $Q_1$.

**Definition 12 (Tuple Arrival Rate).** *The data stream tuple arrival rate, denoted by $\lambda$, is the number of tuples received in a given time instance.*

Intuitively, a higher tuple arrival rate translates into less imprecision as more tuples are available to form equivalence classes with fewer false-positives.

## 3.2 Precision-Bounded Access Control

A precision-bounded access control framework for privacy-preserving data streams is proposed as shown in Fig. 4. The Privacy Protection Mechanism (PPM) ensures that the privacy and precision goals are met before the sensitive stream data is made available to the access control mechanism. The access control policy administrator defines sliding-window queries that define the authorized view of the data stream for each role. The PPM uses generalization of stream data tuples to anonymize and satisfies the given privacy requirement. Generalization adds uncertainty resulting in a reduction of precision of authorized view. The uncertainty due to generalization can be reduced by delaying the stream tuples and forming equivalence classes with less imprecision. However, the delay introduces false-negatives if the stream tuples belonging to the authorized view are held by PPM. The access control policy administrator provides the imprecision bound for each query. PPM needs to ensure that at the time of query evaluation the sum of false-negatives and false-positives is less than the imprecision bound.

The purpose of access control is to ensure that each user accesses only the authorized information. False-positives due to generalization under the *Overlap* semantics imply that access is being provided to unauthorized tuples. False-negatives as a result of *Enclosed* semantics although deny access to the authorized information but ensure that the access control policy is not violated. The reference monitor can be set to enforce an access control policy using either the *Overlap* semantics or strict access control enforcement using the *Enclosed* semantics. Choice of each semantics has its own pros and cons. Relaxed enforcement is beneficial for applications where false alarms are acceptable as compared to the risks associated with a missed event, e.g., epidemic surveillance and airport security. Strict enforcement is suitable for applications where a high risk is associated with a false alarm as compared to a missed event, e.g., false arrest in case of shoplifting. In this paper, the focus is on relaxed enforcement. However the proposed approach for anonymization is also valid for strict enforcement because the heuristic proposed in Section 4 reduces the overlap between anonymized partitions and predicate sliding-window queries. For the *Overlap* semantics, the reference monitor may be set to deny access to a permission if false-positives are more than a desired threshold.

## 3.3 The PACE Problem

The Precision-bounded Access Control for privacy-preserving data strEams (PACE) problem is defined as follows:

**Definition 13 (The PACE Problem).** *Given a data stream $T[i]$, a set of predicate sliding-window queries $Q$, and privacy parameter $k_s$, the Precision-bounded Access Control for privacy-preserving data strEams (PACE) problem is to generate an anonymized stream $T^P[i]$ such that the sum of the average query bound violation for all queries $q \in Q$ is minimized.*

The optimal $k$-anonymity problem based on generalization has been shown to be NP-complete [17]. The hardness result for the PACE problem follows the construction of LeFevre et al. [18] that shows the hardness of $k$-anonymous multi-dimensional partitioning with the smallest average equivalence class size. The decision problem for $k$-anonymous partitioning satisfying the query imprecision bounds for relational data has been shown to be NP-complete [16].

For the decision version of the problem, we consider a single time instance and a set of queries $q \in Q$. The data stream tuples received at this time instance can be transformed into an equivalent set of distinct $(tuple, count)$ pairs. All the queries are evaluated at this time instance. The constant $qv$ defines an upper bound for the sum of the average query bound violation for all predicate sliding-window queries. The tuples received can either be published as partitions or can be put on hold. The decision version of the PACE problem is as follows:

**Definition 14 (The Decisional PACE Problem).** *Given a set $t \in T$ of unique $(tuple, count)$ pairs received at a given time instance and a set of sliding-window queries $q \in Q$ with imprecision bounds $B_q$, does there exist a multidimensional partitioning for $T$ such that every published multidimensional region $R_i$ in $T^p$, $\sum_{t \in R_i} count(t) \geq k_s$ and sum of average query bound violation for all queries is less than the positive constant $qv$?*

**Theorem 3.1.** *Decisional PACE problem is NP-complete.*

**Proof.** Refer to Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2015.2391098. □

### 3.4 Probabilistic Analysis for Query Bound Violation

A precision-bounded access control framework for privacy-preserving data streams has been presented in Section 3.2. The access control policy administrator sets the imprecision bound for each predicate sliding-window query and requires that the imprecision bound for the least number of queries is violated by the PPM. The policy administrator can revise the imprecision bounds for the queries if the probability of satisfying the bound of large number of queries at any time instance is very low. From this perspective, we are interested in the following questions:

- What is the probability that the number of queries violating imprecision bounds is less than a given threshold or is in a given range at any given time instance?
- How long does it take for the sum of average query-bound violation for all queries to reach a steady-state value?

To answer these questions, let $X_1[j], \dots, X_n[j]$ be a set of independent random variables such that $Pr(X_i[j] = 1) = p_i$ and $Pr(X_i[j] = 0) = 1 - p_i$, where $0 \le p_i \le 1$. $X_i[j]$ is equal to 1 if a sliding-window query say, $Q_i$ (size $= w$ and step $=$ 1) evaluated at time instance $j$ violates the imprecision bound, otherwise $X_i[j]$ is equal to 0. The step size for all $n$ queries is 1. Thus, all queries are evaluated at each time instance. The number of queries violating imprecision bounds at time instance $j$ is $X[j] = \sum_{i=1}^{n} X_i[j]$. $X_1[j], \dots, X_n[j]$ at each time instance are modeled as *Poisson trials* and follow a *Poisson binomial* distribution. The expected number of queries violating the imprecision bound at time instance $j$ is $E[X[j]] = \mu = \sum_{i=1}^{n} p_i$ [19]. Dependency may exist among the sliding-window queries evaluated at each time instance but for our analysis we assume that they are independent.

By the law of large numbers, the difference between the actual and expected values for a random process decreases as the number of trials increases. Formally, for a set of independent non-identically distributed random variables $X_1[j], \dots, X_n[j]$ and $X[j] = \sum_{i=1}^{n} X_i[j]$, the sample average $\bar{\mu} = \frac{1}{j}\sum_{i=1}^{j} X[i]$ at time $j$, converges to the expected value $\mu = E[X[j]]$ as $j$ approaches $\infty$ [20]. The sample average $\bar{\mu}$ for a large number of samples can be used to answer the first question: What is the probability that the number of sliding-window queries violating bounds is less than a given threshold? We use the Hoeffding/Chernoff bound [21] for the *Poisson trials* as given in Lemma 3.2.

**Lemma 3.2.** *Let* $X_1[j], \dots, X_n[j]$ *be an independent* Poisson trial *at time instance* $j$*, then for* $X[j] = \sum_{i=1}^{n} X_i[j]$*,* $\mu = E[X[j]]$*, and* $0 < \epsilon \le 1$*, we have*

$$Pr[X[j] < (1-\epsilon)\mu] < e^{-\frac{\mu\epsilon^2}{2}}. \tag{5}$$

The $\epsilon$ value is set according to the required threshold. However, in order to use Lemma 3.2 we would like to know the sample size that gives a high probability of $|\bar{\mu} - \mu|$ being smaller than some constant $x$. Theorem 3.3 provides a lower bound on $S$, the sample size, for a given probability that $|\bar{\mu} - \mu| \ge x$. The proof for Theorem 3.3 is similar to the proof of generalized pairwise-independent sampling theorem [22].

**Theorem 3.3.** *Let* $X_1[j], \dots, X_n[j]$ *be an independent* Poisson trial *at time instance* $j$*.* $X_i[j]$ *is a random variable that is 1 if a sliding-window query say,* $Q_i$ *(with size* $= w$ *and step* $= 1$*) that is evaluated at time instance* $j$ *violates the imprecision bound, otherwise* $X_i[j]$ *is 0. The* $Var[X_i] \le b$ *for* $b \ge 0$*,* $X[j] = \sum_{i=1}^{n} X_i[j]$*,* $\bar{\mu} = \frac{1}{S}\sum_{i=1}^{S} X[i]$*,* $\mu = E[X[j]]$*, and* $x > 0$*, we have*

$$S \ge \frac{bn}{Pr[|\bar{\mu} - \mu| \ge x]x^2}. \tag{6}$$

**Proof.** Refer to Appendix A, available in the online supplemental material. □

**Example 5.** Suppose that for 500 predicate sliding-window queries (i.e., n = 500) the tolerance for $\bar{\mu}$ is $x = 2.5$ and we want our estimate to be within this tolerance with a 95 percent probability. The Var$[X_i]$ is equal to $p_i(1-p_i)$ and the maximum value $b$ of $p_i$ in the interval $0 \le p_i \le 1$ occurs for $p_i = \frac{1}{2}$. From Equation (6), we get a sample size of 800, which implies that when 800 time-stamps have elapsed, there is a 95 percent probability that $|\bar{\mu} - \mu| < 2.5$.

According to central limit theorem the *Poisson binomial* distribution for a large number of samples can be approximated by a normal distribution of sample mean $\bar{\mu}$ and standard deviation $\bar{\sigma}$ [23]. The cumulative distribution function of the approximate normal distribution can then be used to find the probability of the number of queries violating the imprecision bounds in a given range at any time instance.

## 4 Algorithm for Precision-Bounded Anonymization

Cao et al. [7] have proposed a clustering algorithm for anonymization of a data stream. Another approach proposed by Zhou et al. [8] uses an $R$-tree [24] based algorithm to anonymize. The stream tuples are added to leaf nodes in an $R$-tree with a constraint that each node should have between $k_s$ to $2k_s$ tuples. When a leaf node is published, that node is removed from the $R$-tree. The proposed heuristic listed in Algorithm 1 can be applied to both techniques for a given predicate sliding-window query workload. We follow the approach suggested by Zhou et al. but use an $R^+$-tree [25] instead of an $R$-tree. The $R^+$-tree-based anonymization algorithm for relational data has been proposed by Iwuchukwu et al. [26]. When the tuples are added to an $R^+$-tree, the leaf nodes and intermediate nodes are non-overlapping [26]. This condition can be maintained for a data streams under the assumption that the identity value of stream tuple is not repeated within the maximum delay. However, if the tuple identity value is repeated, then according to the data stream $k_s$-anonymity definition, the tuple cannot be put into the same leaf node. Adding that tuple to any other leaf node creates an overlapping leaf node. It is observed in the experiments in Section 5 that better accuracy can be achieved by maintaining non-overlapping leaf nodes and by setting the maximum delay such that the existing leaf nodes are published before any duplicate tuple is received.

An $R^+$-tree-based index is maintained by the PPM. The data stream tuples in $T[i]$ are first added to the active $R^+$-tree at each time instance. Then, the decision to publish each leaf node (equivalence class) is taken. For a leaf node in the active $R^+$-tree, the expected false-positives ($EFP$) and the expected false-negatives ($EFN$) are defined as follows.

**Definition 15 (Expected False-Positives).** *The Expected False-Positives for a leaf-node Partition $P$ ($EFP_P$, for short) is defined as the sum of false-positives for all queries resulting from Partition $P$, provided the partition is published at the current time instance*

$$EFP_P = \sum_{Q_j \in Q} |P - Q_j|. \tag{7}$$

In Equation (7) above, the minus sign denotes the set difference operation which gives the data stream tuples that are inside the partition but are outside the region defined by the predicate sliding-window query $Q_j$. If Partition $P$ is published then all these tuples satisfy Definition 7 for a false-positive tuple.

**Definition 16 (Expected False-Negatives).** *The Expected False-Negatives for a leaf-node Partition $P$ ($EFN_P$, for short) is defined as the sum of false-negatives for all queries that are evaluated at the next time instance resulting from Partition $P$ if the partition is held by the PPM at the current time instance.*

$$EFN_P = \sum_{Q_j \in Q} |Q_j(P)| \tag{8}$$

In Equation (8) above, only those sliding-window queries that add false-negatives are evaluated in the next time instance. The tuples inside the Partition $P$ that are also inside the region defined by a predicate sliding-window Query $Q_j$ satisfy Definition 8 for a false-negative tuple.

A leaf-node in the $R^+$-tree can either add false-positives (if published) or false-negatives (if held) towards the sliding-window queries. The false-positives represent the information loss due to generalization while false-negatives represent the information loss due to publishing delay. Therefore, we choose the option that contributes less imprecision for all the queries with respect to a partition. In other words, a leaf-node partition can be held in the active $R^+$-tree until $EFP_P$ is smaller than $EFN_P$. We also define $w_{FP}$ and $w_{FN}$ as weights where $0 < w_{FP}, w_{FN} \leq 1$. The weight assignment should be done according to requirements of the application, e.g., $w_{FP}$ can be set less than 1 for an application sensitive to false-negatives.

The Total Imprecision Minimization (TIM) algorithm is listed in Algorithm 1. The active $R^+$-tree is initialized in Line . In the **for** loop in Lines 2-5, at each time instance, the tuples arriving in data stream $T[i]$ are added to a leaf node in the active $R^+$-tree and the leaf node is split if the size of new leaf nodes is greater than $k_s$. The leaf node is split along the median in the dimension having the least expected false-positives. The **for** loop in Lines 6-9 checks all the leaf nodes of the active $R^+$-tree. If the expected false-negatives by holding a leaf node are more than the false-positives as a result of publishing the node, then the node is published as an equivalence class. For $w_{FP} = w_{FN}$, the sum of false-negatives for all queries is always greater than total false-positives because until a partition is published, it only adds false-negatives. Access control enforcement under the *Overlap* semantics can be adjusted to set a preference for lower false-positives ($w_{FP} > w_{FN}$) or lower false-negatives ($w_{FN} > w_{FP}$). The same algorithm can be used to satisfy the privacy requirements of $l_s$-diversity (Listing 2 in Appendix B, available in the online supplemental material) or variance diversity by splitting the leaf node in Line of Algorithm 1, provided both the new leaf nodes meet the privacy requirement.

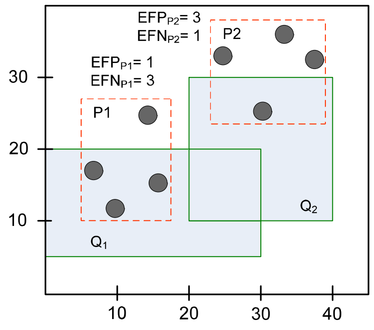


Fig. 5. $EFP_P$ and $EFN_P$ for leaf nodes $P1$ and $P2$.

**Algorithm 1.** Total Imprecision Minimization

```
Input: $T[i]$, $k_s$, $Q$, and $B_{Q_j}$
Output: $EC_1, EC_2, \ldots$
Initialize the active $R^+$-tree;
2: for (each $t_m \in T[i]$ arriving at time instant $i$) do
3:    Add Tuple $t_m$ to a leaf node in active $R^+$-tree;
4:    if (Size of new leaf nodes after splitting is $> k_s$) then
5:       Split the leaf node;
6: for (all leaf nodes $P$ in active $R^+$-tree at time instant $i$) do
7:       Update the imprecision cost of each leaf node;
8:       if (($w_{FN} * EFN_P > EFP_P * w_{FP}$) OR (($i$ - $t_m$.TS) $\geq$ ($\delta$ - 1)))
         then
9:             Publish the leaf node as $EC$ and remove from active
               $R^+$-tree;
```

**Example 6.** Assume that eight tuples are received at some time instance as shown in Fig. 5. The shaded rectangles with solid lines represent sliding-window queries while the rectangles with dotted lines represent partitions. The leaf-node Partitions P1 and P2 are added to the $R^+$-tree. The weight assignment is $w_{FP} = w_{FN} = 1$. The two sliding-window queries $Q_1$ and $Q_2$ are to be evaluated at the next time instance. The EFP for P1 is 1 and EFN is 3. Since EFP is less than EFN for P1, P1 is published. For P2, EFP is 3 and EFN is 1. Since EFN is less than EFP for P2, P2 is put on hold by the PPM. The total imprecision contributed by P1 and P2 at this time instance is 2. Note that if both P1 and P2 are published or held by PPM the total imprecision will be 4.

## 5 EXPERIMENTS

The experiments for the empirical evaluation of the proposed algorithm have been carried out on two real datasets.

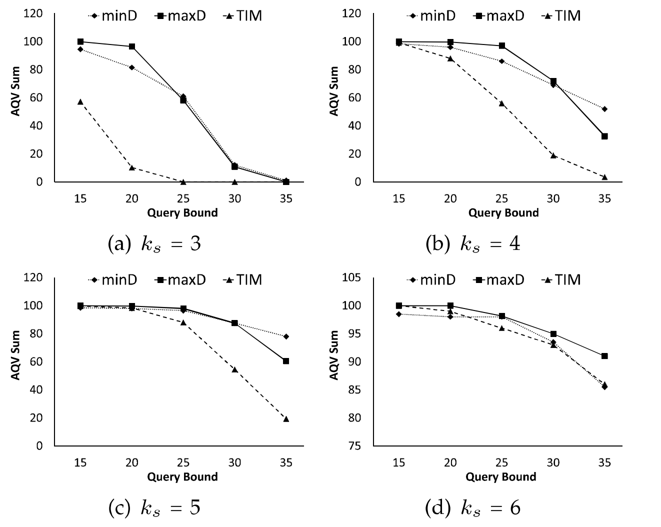


Fig. 6. The sum of the average query-bound violation for $k_s$-anonymity for the Adult dataset.

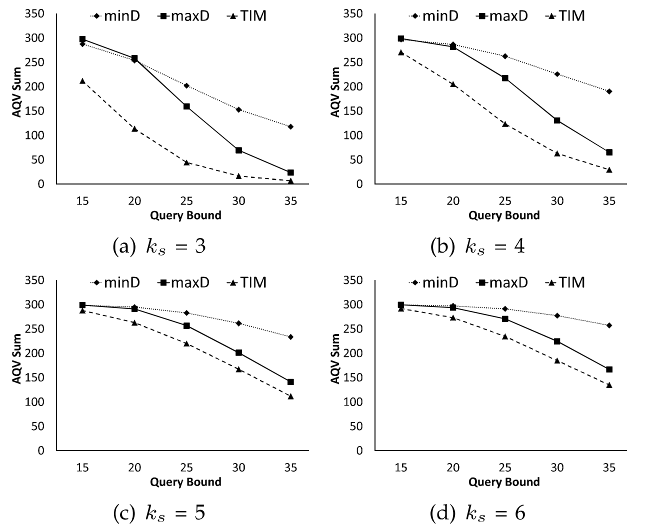


Fig. 7. The sum of the average query-bound violation for $k_s$-anonymity for the Census dataset.

The first dataset is the Adult dataset from UC Irvine Machine Learning Repository [27] having 45,222 tuples and is the benchmark for $k$-anonymity research. The attributes in the Adult dataset are: Age, Work class, Marital status, Relationship, Race, Gender, Education, and, Occupation. The second dataset is the Census dataset [28] from IPUMS.[1] This dataset is extracted for the year 2001 using attributes: Age, Gender, Marital status, Race, Language, Education, Occupation, and Income. The size of this dataset is about 1.2 million tuples. For the $k_s$-anonymity experiments, we use the first six attributes as quasi-attributes from both datasets. To model the dataset as a data stream, we assume that 1,000 tuples are received at each time instance. The maximum delay constraint $\delta$ is set to five time units for the Adult dataset and 25 time units for the Census dataset. It is assumed that the time interval between the two time instances is enough to update the $R^+$-tree and the query imprecision at each time instance. We also assume that the tuple ID is not repeated within the time duration of the maximum delay.

We use 100 and 300 queries as the workload/permissions for the Adult and Census datasets, respectively. The queries are generated randomly using the approach suggested by Iwuchukwu, et al. [26]. In this approach, two tuples are selected randomly from the tuple space and a query is formed by making a bounding box of these two tuples. The bounding box gives the predicates for the sliding-window query. The window size and step are also selected randomly from a fixed range. For the Adult dataset, the range for the window size is 20-30 and for the step the range is 10-20. For the Census dataset, the range for the window size is 100-200 and for the step is 50-100. The random query is then added to the workload if the sliding-window query meets the size constraint for the first step (8,000 for the Adult dataset and 15,000 for the Census dataset). The imprecision bounds for all sliding-window queries are set based on the query size at the time of query evaluation. For example, an imprecision bound of 10 percent for a sliding-window query implies that, at each step, when the query is evaluated, the imprecision should be less than 10 percent of the query size at that time instance.

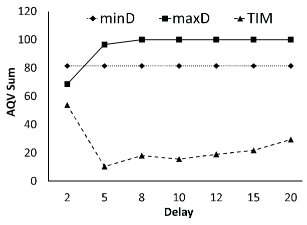
The approach proposed by Cao et al. [7] for data stream anonymization tries to minimize the error due to generalization with a constraint that tuples must be published before the maximum delay deadline. Zhou et al. [8] propose an $R$-tree-based approach to anonymize the data stream and propose a minimum-delay heuristic, where tuples are published as soon as they meet the privacy condition. They also propose a randomized algorithm to minimize a delay-based cost function and show that the accuracy can be further improved by taking the tuple distribution into account. In our experiments, we compare the proposed approach (i.e., Total Imprecision Minimization) with the maximum delay heuristic (denoted by maxD) and the minimum delay publishing (denoted by minD). $w_{FP}$ and $w_{FN}$ are set to 1 for TIM.

## 5.1 Varying Imprecision Bound

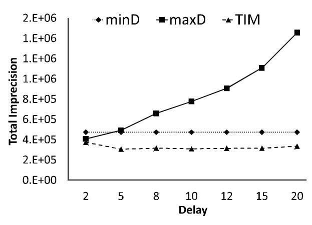
For the $k_s$-anonymity experiments, the value of $k_s$ is fixed and the query imprecision bound is varied from 15 to 35 percent with increments of 5 and the sum of the average query-bound violation for all predicate sliding-window queries is evaluated. The results for $k_s$-anonymity are given in Fig. 6 for the Adult dataset for $k_s$ values of 3, 4, 5 and 6. The minimum delay heuristic is better than the maximum delay heuristic but TIM gives the best results for all values of $k_s$.

For the Census dataset results for $k_s$-anonymity are given in Fig. 7 for $k_s$ values of 3, 4, 5 and 6. The performance of TIM is better than that of minD and maxD for all values of $k_s$ in the Census data. In this case, maxD performs better than minD as compared to the Adult dataset. The performance of maxD is dependent upon the maximum delay value. In the case of the Adult dataset, maxD performs better than minD only when the delay value is 2 as given in Fig. 8a but the experiments were performed with $\delta = 5$. For the Census dataset with $\delta = 25$, it can be observed in Fig. 9a that the maxD heuristic is better than minD heuristic.

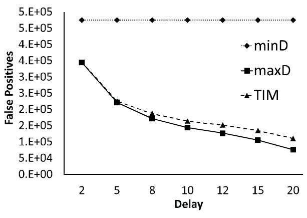
1. Available at http://usa.ipums.org/usa/



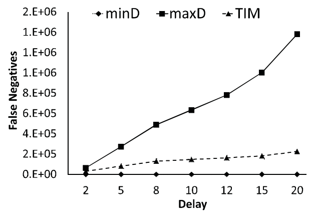
(a) Average query-bound violation



(b) Total imprecision



(c) False-positives



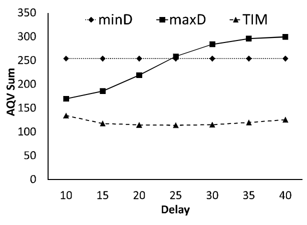
(d) False-negatives

Fig. 8. Varying the maximum delay ($\delta$) parameter for the Adult dataset.

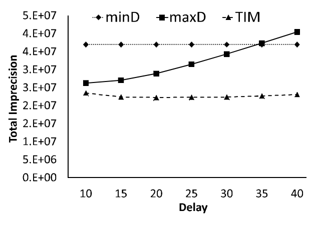
## 5.2 Varying the Maximum Delay ($\delta$) Parameter

In the next experiment for the Adult dataset, $k_s$ is set to 3 and the query imprecision bound is set to 20 percent of the sliding-window query size at the time of query evaluation. The maximum delay value ($\delta$) is varied from 2 time units to 20 time units as given in Fig. 8. The total imprecision in Fig. 8b is the sum of false-positives in Fig. 8c and false-negatives in Fig. 8d. The total imprecision, false-positives, and false-negatives are calculated by adding the imprecision values for all queries. The average query violation and the total imprecision for the minimum delay heuristic remain constant for all values of the $\delta$ as they are independent of the delay value. For the minimum delay heuristic, the false-negatives are zero as given in Fig. 8d because the partitions are published without delay in the same time instance. TIM has the best performance in terms of the average query-bound violation and the total imprecision as noted in Fig. 8a and Fig. 8b. We can notice in Fig. 8c that as the $\delta$ value is increased, the total false-positives decrease for TIM as more data stream tuples are available to form partitions with less imprecision. Also, we can observe in Fig. 8d that as the $\delta$ value is increased the false-negatives increase because more tuples are put on hold by the PPM.

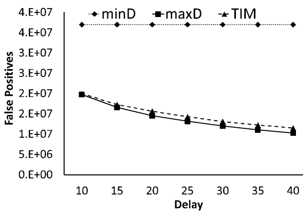
For the Census dataset, $k_s$ is fixed at 3 and query imprecision bound is fixed at 20 percent of the sliding-window query size at the time of query evaluation. Then, the maximum delay ($\delta$) value is varied from 10 time units to 40 time units in increments of 5 time units as given in Fig. 9. It can be observed that TIM has the best performance in terms of the average query-bound violation and the total imprecision as given in Figs. 9a and 9b. For TIM, as the $\delta$ is increased, the false-positives decrease as given in Fig. 9c because more data stream tuples are available to form partitions with fewer false-positives. On the other hand, the false-negatives increase as given in Fig. 9d with the increase in the $\delta$ value as more tuples are held by PPM.



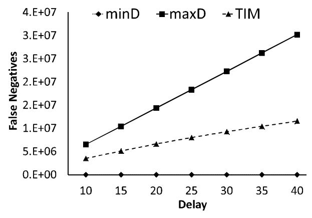
(a) Average query-bound violation



(b) Total imprecision

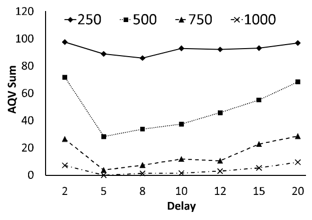


(c) False-positives

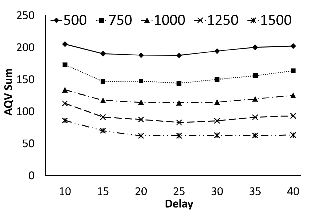


(d) False-negatives

Fig. 9. Varying the maximum delay parameter ($\delta$) for the Census dataset.



(a) Adult dataset



(b) Census dataset

Fig. 10. Varying the rate of tuple arrival for TIM.

## 5.3 Varying the Rate of Tuple Arrival ($\lambda$) for TIM

The next experiment is performed to analyze the effect of the rate of tuple arrival on TIM. Intuitively, the higher rate of tuple arrivals should give better results because more data stream tuples are available at a given time instance to form partitions with fewer false-positives. For this experiment, we vary tuple arrival rates as 250, 500, 750, and 1,000 tuples for Adult dataset at each time instance, a $k_s$ value of 3 and an imprecision bound of 25 percent. We can notice in Fig. 10a that, as the tuple arrival rate is increased, the sum of average query-bound violations decreases for all maximum delay ($\delta$) values.

For the Census dataset the same trend as that of the experiment on the Adult dataset is observed. The performance improves as the rate of tuple arrival increases. For this experiment, we vary the tuple arrival rates as 500, 750, 1,000, 1,250, and 1,500 tuples for each time instance, a $k_s$ value of 3 and an imprecision bound of 20 percent. From Fig. 10b, we can observe that as the tuple arrival rate is increased the performance is improved because more data stream tuples are available to form partitions with fewer false-positives.

## 5.4 Varying Weights ($w_{FN}, w_{FP}$) for TIM

In TIM, a partition is published when the expected false-negatives are greater than the expected false-positives. In this experiment for the Adult dataset, weights are assigned to expected false-negative and false-positive values. By TIM14, it is meant that weight $w_{FN}$ is four times higher than the weight $w_{FP}$ (and vice versa for TIM41). It can be observed in Fig. 11a, that for TIM14, there are more false-positives than for TIM and in Fig. 11b, there are less false-negatives as partitions are published early. Similarly, for TIM41 as compared to TIM, the false-positives are less and

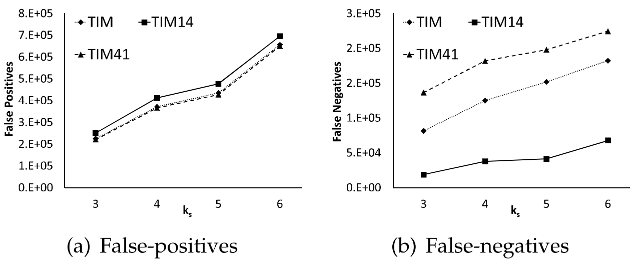


Fig. 11. The imprecision for TIM using weights for the Adult dataset.

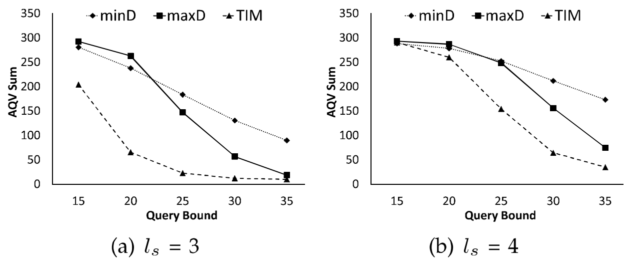


Fig. 13. $l_s$-diversity for the census dataset.

the false-negatives are more as partitions are held for a longer time by PPM.

## 5.5 Duplicate Tuples Received within the Maximum Delay ($\delta$)

In the previous experiments, it is assumed that the stream tuple-id values are not repeated. We now assume that a tuple-id can be repeated only once within the maximum delay ($\delta$) after time $\frac{\delta}{2}$. The $\delta$ is set to 10 and the total imprecision is plotted for various values of $k_s$. The repetition of tuple-id forces PPM to create overlapping leaf nodes in the $R$-tree to satisfy the $k_s$-anonymity requirement. The results of the experiment shown in Fig. 12a reveal that maxD has the worst performance for all the values of $k_s$ because more delay allows more overlapping leaf nodes while minD has the best performance. For TIM2 and maxD2, knowing that a tuple-id is repeated after time $\frac{\delta}{2}$, we set the $\delta$ to 5 and use an $R^+$-tree with non-overlapping leaf nodes. This allows to reduce the total imprecision by an order of 3 as shown for TIM2 in contrast with minD.

We repeat this experiment for the Census dataset with a $\delta$ value of 20. Notice that in Fig. 12b, the performance of minD is better than maxD and TIM due to fewer overlapping leaf nodes. For TIM2 and maxD2, knowing that the tuple-id is repeated after time $\frac{\delta}{2}$, the $\delta$ value is set to 10 and an $R^+$-tree is used with non-overlapping leaf nodes. This allows to reduce the imprecision by an order of 3 for TIM2.

## 5.6 $l_s$-Diversity and Stream Variance Diversity

We use Attribute *occupation* as the sensitive attribute and the first six attributes as the QI attributes for the $l_s$-diversity experiments on data streams using the Census dataset. All the tuples having the occupation value as Not Applicable (0 in the dataset) in the Census dataset are removed leaving about 700 k tuples. The proposed algorithm in Listing 2 (Refer to Appendix B, available in the online supplemental material) is used for $l_s$-diversity with the constraint that each leaf node should be $l_s$-diverse after splitting. The experiment is conducted for the $l_s$ values of 3 and 4. For each value of $l_s$, we vary the query imprecision bounds from 15 to 35 percent with increments of 5 and find the sum of average query-bound violation for all sliding-window queries. The results are given in Fig. 13 and demonstrate that TIM has the lowest average query-bound violation for $l_s$-diversity.

For the data stream variance diversity experiments, we use Attribute *income* as the sensitive attribute. All the tuples having the income value as Not Applicable (9,999,999 in the dataset) in the Census dataset are removed, which leaves about 950 k tuples. The experiments are conducted for the variance values $\frac{V}{200}$ and $\frac{V}{100}$, where $V$ is the variance of the sensitive attribute in the dataset. For a given variance diversity value , the query imprecision bound is changed from 15 to 35 percent and the sum of the average query-bound violation for each publishing approach is calculated. Similar to $k_s$-anonymity and $l_s$-diversity, each leaf node in the active $R^+$-tree needs to satisfy the variance diversity condition. The results for data stream variance diversity are given in Fig. 14 and illustrate that TIM gives the best results as compared to minD and maxD.

## 5.7 Sample Size for Sample Mean to Stabilize

In this experiment, the number of sliding-window queries violating bounds for TIM along with the sample mean (average query-bound violation) is given in Fig. 15. We randomly select 500 queries (with window size = 25 and step = 1) with cardinality $\geq$ 2500 in the first step and set imprecision bound to 30 percent of the query size. One of the important observations in this plot is the dependence of *Poisson trials* on query window size. The number of sliding-window queries violating bounds changes abruptly after intervals of about 25 time units. The dependence among trials decreases if the query window size is reduced. Secondly, in Example 5, the sample size for tolerance $|\bar{\mu} - \mu| < 2.5$ with a 95 percent probability is found to be greater than 800. Observe that in Fig. 15, after 800 timestamps, $\bar{\mu}$ is almost stable.

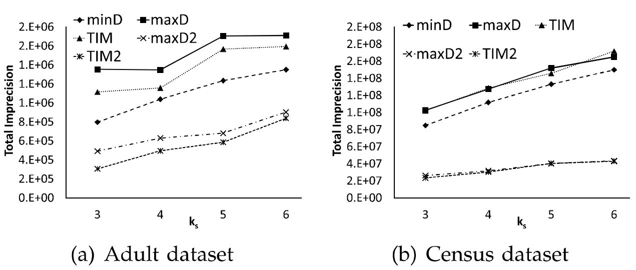


Fig. 12. Duplicate tuples received within the maximum delay ($\delta$).

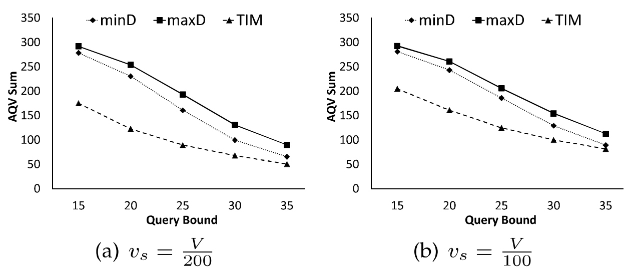


Fig. 14. Stream variance diversity for the census dataset.

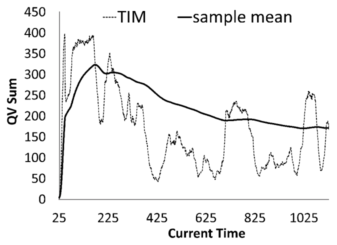


Fig. 15. Average query-bound violation versus Time for the Census dataset

## 5.8 Visual Representation of Heuristics

The visual representations of the published partitions resulting from the approaches minD, maxD, TIM with $R$-tree, and TIM with $R^+$-tree are given in Fig. 16. 1,200 tuples with two attributes are randomly selected (using a Normal distribution with $\mu = 50$, $\sigma = 10$, and cardinality = 100) and it is assumed that rate of tuple arrival $\lambda = 200$. Five random predicate sliding-window queries are selected (the query size is greater than 400) with window size 5 and window step 2. The query imprecision bound is set to 15 percent of the sliding-window query size at the time of query evaluation. The maximum delay $\delta$ is set to 3 time-units. The rectangles with the blue (darker) lines are the queries while the rectangles with the red (lighter) lines are the partitions generated by the heuristics at $k_s = 3$. The partitions held by PPM are given in green (very light) color and the false-negative tuples inside these rectangle are marked by $*$ while the tuples outside the queries are marked by $\times$.

The summary of the comparisons for minD, maxD, TIM with $R$-tree, and TIM with $R^+$-tree is given in Table 1. In this table, AQV stands for the Average Query-bound Violation. Minimum delay publishing has zero false-negatives but the highest false-positives as shown in Fig. 16a. Observe that in Fig. 16b, maximum delay publishing allows to reduce the false-positive but considerably increases the number of false-negatives. In comparison, the partitions published by TIM given in Fig. 16c have the lowest total imprecision and violate the bounds for the minimum number of predicate sliding-window queries. Fig. 16d gives the partitions published by TIM using an $R$-tree instead of an $R^+$-tree. It can be noted that the $R$-tree approach creates overlapping leaf nodes resulting in higher false-positives.

## 6 Related Work

In this section, first the literature related to access control on data streams is reviewed and then research related to privacy preserving publishing of data streams is discussed. To the best of our knowledge both the precision-bounded access control and privacy together for data streams have not been investigated before.

Nehme et al. propose security punctuation-based access control framework for data streams [3]. A security punctuation is a predicate that defines access to stream data and is created by the user generating stream data. The security punctuation tuples are then interleaved in the data stream. The subjects are assigned roles on the server and can execute authorized queries on the incoming data stream. The

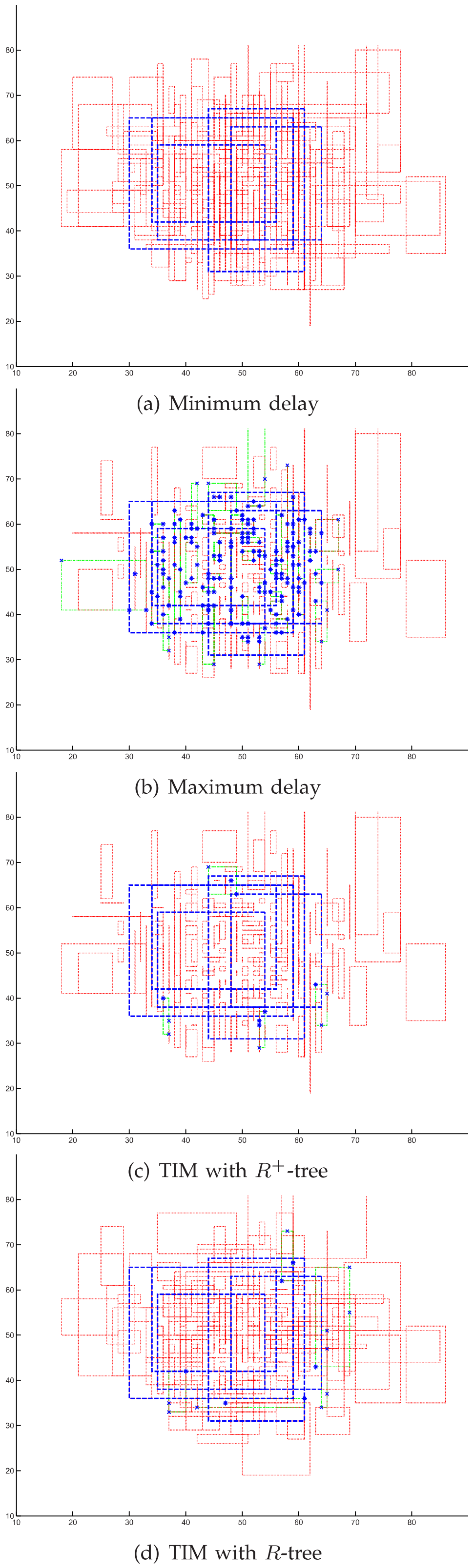


Fig. 16. Anonymization for two attributes with discrete normal distribution ($\mu = 50, \sigma = 10$).

server allows the roles access to stream tuples according to the embedded security punctuation.

Role-based access control for data streams has been proposed by Carminati et al. [4]. In their framework, there are two types of temporal constraints. First is the interval

TABLE 1
Comparison of Heuristics

| | Total FP | Total FN | FP+FN | AQV |
|---|---|---|---|---|
| **minD** | 403 | 0 | 403 | 3 |
| **maxD** | 126 | 595 | 721 | 5 |
| **TIM($R^+$-tree)** | 189 | 12 | 201 | 0 |
| **TIM($R$-tree)** | 328 | 10 | 338 | 4 |

constraint during which the role can access stream data. Second is the window constraint that limits access to the data stream for each role according to the authorized view defined by the sliding-window query predicate. They consider two types of privileges over the authorized data that is read privilege for selection and projection operations and aggregate privilege for Min, Max, Count, Avg, and Sum operations. In the current paper, we follow the access control specification of Nehme et al. and Carminati et al. but further consider the privacy-preservation along with the precision-bounded access control.

Cao et al. have proposed CASTLE for continuously anonymizing data streams [7]. They extend the definition of $k$-anonymity for data streams and propose a clustering algorithm that publishes anonymized clusters before a given maximum delay deadline. The measure used to assess the quality of published clusters is the information loss metric that does not consider the loss due to delay in publishing. To overcome this shortcoming, Zhou et al. proposed a delay-based anonymization quality measure that increases the information loss as the publishing delay increases [8]. They propose a randomized-algorithm based on the R-tree. The data stream tuples are added to the active R-tree and the leaf nodes of the tree due at each time instance are published. The due time for each node is evaluated randomly based on the information loss. They further use the distribution density of the data stream to improve the algorithm. Both Cao et al. and Zhou et al. suppress the time-stamp attribute in the anonymized stream. However, the time-stamp attribute is required to evaluate any sliding-window query over the anonymized stream.

Dwork et al. have proposed *differential privacy* for data streams considering a single aggregate query [29]. Cao et al. further extend the model to sliding-window queries over binary data streams [30]. *Differential privacy* is achieved by adding random noise to original query results and offers better privacy guarantees than generalization, however syntactic anonymization techniques (e.g., generalization) provide better precision [31]. The *Differential privacy* model for relational data streams is still to be developed. In the current paper our focus is on generalization and we introduce precision bounds for sliding-window queries over privacy-preserving data streams.

Access control and privacy techniques have been investigated for static relational data. LeFevre et al. [10], [18] and Iwuchukwu et al. [26] have proposed workload-aware anonymization for micro data publishing. Work has been done on micro data anonymization with accuracy and privacy constraints [16], [32], [33]. We have proposed the concept of imprecision bounds for accuracy-constrained access control on relational data [16]. However, the access control on data streams presents different challenges because of the temporal constraints defined by sliding-window query predicates. In the data stream literature, access control and privacy-preserving publishing have been considered in isolation. However, we propose a unified precision-bounded access control framework for privacy-preserving data streams.

## 7 Conclusions

In this paper precision-bounded access control for privacy-preserving data streams has been proposed. The access control administrator defines the permitted view of the data stream along with the required precision. The privacy protection mechanism applies generalization to the stream data such that the privacy requirement is met and imprecision bound for the maximum number of sliding-window queries is satisfied. An algorithm has been proposed to minimize the total imprecision and experiments have been performed to compare the performance. In future work, we plan to extend the access control enforcement to *Enclosed semantics*. Also, in this paper, syntactic anonymization techniques have been considered. We plan to extend the *differential privacy* model for sliding-window queries over binary data streams [30] proposed by Cao et al. to relational data streams.

## Acknowledgments

The research has been supported by NSF under Grants IIS-1117766 and IIS-0964639. Zahid Pervaiz is the corresponding author.

## References

[1] L. Golab and M. Özsu, “Issues in data stream management,” *ACM Sigmod Rec.*, vol. 32, no. 2, pp. 5–14, 2003.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, “Models and issues in data stream systems,” in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst.*, 2002, pp. 1–16.

[3] R. Nehme, E. Rundensteiner, and E. Bertino, “A security punctuation framework for enforcing access control on streaming data,” in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 406–415.

[4] B. Carminati, E. Ferrari, J. Cao, and K. Tan, “A framework to enforce access control over data streams,” *ACM Trans. Inf. Syst. Security*, vol. 13, no. 3, p. 28, 2010.

[5] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 6, pp. 1010–1027, Nov. 2001.

[6] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 3, 2007.

[7] J. Cao, B. Carminati, E. Ferrari, and K. Tan, “Castle: Continuously anonymizing data streams,” *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 99, pp. 337–352, May/Jun. 2011.

[8] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia, “Continuous privacy preserving publishing of data streams,” in *Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol.*, 2009, pp. 648–659.

[9] J. Buehler, A. Sonricker, M. Paladini, P. Soper, and F. Mostashari, “Syndromic surveillance practice in the united states: Findings from a survey of state, territorial, and selected local health departments,” *Adv. Disease Surveillance*, vol. 6, no. 3, pp. 1–20, 2008.

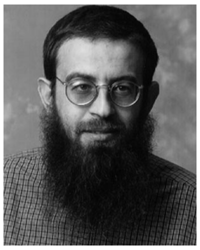
[10] K. LeFevre, D. DeWitt, and R. Ramakrishnan, “Workload-aware anonymization techniques for large-scale datasets,” *ACM Trans. Database Syst.*, vol. 33, no. 3, pp. 1–47, 2008.

[11] N. Li, T. Li, and S. Venkatasubramanian, “Closeness: A new privacy measure for data publishing,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 7, pp. 943–956, Jul. 2010.

[12] T. Ghanem, A. Elmagarmid, P. Larson, and W. Aref, "Supporting views in data stream management systems," *ACM Trans. Database Syst.*, vol. 35, no. 1, p. 1, 2010.
[13] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proc. 2nd Int. Conf. Mobile Data Manag.*, 2001, pp. 3–14.
[14] J. Gehrke, F. Korn, and D. Srivastava,, "On computing correlated aggregates over continual data streams," *ACM SIGMOD Rec.*, vol. 30, no. 2, pp. 13–24, 2001.
[15] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Security*, vol. 4, no. 3, pp. 224–274, 2001.
[16] Z. Pervaiz, W. G. Aref, A. Ghafoor, and N. Prabhu, "Accuracy-constrained privacy-preserving access control mechanism for relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 795–807, Apr. 2014.
[17] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation algorithms for k-anonymity," *J. Privacy Technol.*, 2005, http://ilpubs.stanford.edu:8090/645/1/2004-24.pdf
[18] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multi-dimensional k-anonymity," in *Proc. 22nd Int. Conf. Data Eng.*, 2006, p. 25.
[19] W. Hoeffding, "On the distribution of the number of successes in independent trials," *Ann. Math. Statist.*, vol. 27, no. 3, pp. 713–721, 1956.
[20] W. Feller, "The strong law of large numbers," in *An Introduction Probability Theory Its Applications*. New York, NY, USA: Wiley, pp. 243–245, vol. 1, 1968.
[21] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.
[22] A. Meyer, "Deviation from the mean," 6.042J / 18.062J Mathematics for Computer Science, Course Notes for Spring, MIT OpenCourseWare, 2010.
[23] W. Feller, "The fundamental limit theorems in probability," *Bull. (New Ser.) Amer. Math. Soc.*, vol. 51, no. 11, pp. 800–832, 1945.
[24] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Readings in Database Systems*, 3rd ed. Cambridge, MA, USA: MIT Press, 1998, pp. 90–100.
[25] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The r+-tree: A dynamic index for multi-dimensional objects," in *Proc. 13th Int. Conf. Very Large Data Bases*, 1987, pp. 507–518.
[26] T. Iwuchukwu, "Anonymization techniques for large and dynamic data sets," PhD thesis, Univ. Wisconsin-Madison, Madison, WI, USA, 2008.
[27] K. Bache and M. Lichman, "UCI machine learning repository," School of Information and Computer Sciences, University of California, Irvine, 2013, http://archive.ics.uci.edu/ml
[28] S. Ruggles, J. T. Alexander, K. Genadek, R. Goeken, M. B. Schroeder, and M. Sobek, "Integrated public use microdata series: Version 5.0 [machine-readable database]," Univ. Minnesota, Minneapolis, MN, USA, 2010.
[29] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proc. 42nd ACM Symp. Theory Comput.*, 2010, pp. 715–724.
[30] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan, "Efficient and accurate strategies for differentially-private sliding window queries," in *Proc. 16th Int. Conf. Extending Database Technol.*, 2013, pp. 191–202.
[31] C. Clifton and T. Tassa, "On syntactic anonymity and differential privacy," in *Proc. IEEE Int. Conf. Data Eng. Workshop Privacy-Preserving Data Publication Anal.*, 2013, pp. 88–93.
[32] S. Chaudhuri, R. Kaushik, and R. Ramamurthy, "Database access control & privacy: Is there a common ground?" in *Proc. 5th Biennial Conf. Innovative Data Syst. Res.*, 2011, pp. 96–103.
[33] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "A framework for efficient data anonymization under privacy and accuracy constraints," *ACM Trans. Database Syst.*, vol. 34, no. 2, p. 9, 2009.

**Zahid Pervaiz** is currently working toward the PhD degree in the School of Electrical and Computer Engineering, Purdue University. His research interests are in data privacy, distributed system security, and access control.

**Arif Ghafoor** is a professor in the School of ECE, Purdue University. His research interests include database security and multimedia systems. He is a fellow of the IEEE.



**Walid G. Aref** is a professor of computer science at Purdue University. His research interests are in developing database technologies for emerging applications, e.g., spatial, multimedia, genomics, and sensor-based databases. He is a senior member of the IEEE.