Privacy-preserving Access Control

Zahid Pervaiz, Walid G. Aref *Senior Member, IEEE*, Arif Ghafoor *Fellow, IEEE*,
Nagabhushana Prabhu
E-mail: zpervaiz@purdue.edu

Abstract—Access control mechanisms protect sensitive information from unauthorized users. However, when sensitive information is shared and a Privacy Protection Mechanism (PPM) is not in place, an authorized insider can still compromise the privacy of a person leading to identity disclosure. A PPM can use suppression and generalization to anonymize and satisfy privacy requirements, e.g., k-anonymity and l-diversity, against identity and attribute disclosure. However, the protection of privacy is achieved at the cost of precision of authorized information.

In this paper, we propose a privacy-preserving access control framework. The access control policies define selection predicates available to roles while the privacy requirement is to satisfy the k-anonymity or l-diversity. An additional constraint that needs to be satisfied by the PPM is the imprecision bound for each selection predicate. The techniques for workload-aware anonymization for selection predicates have been discussed in the literature. However, to the best of our knowledge, the problem of satisfying the accuracy constraints for multiple roles has not been studied before. In our formulation of the aforementioned problem, we propose heuristics for anonymization algorithms and show empirically that the proposed approach satisfies imprecision bounds for more permissions and has lower total imprecision than the current state of the art.

Index Terms—Access Control, Privacy, k-anonymity, Query Evaluation.

1 Introduction

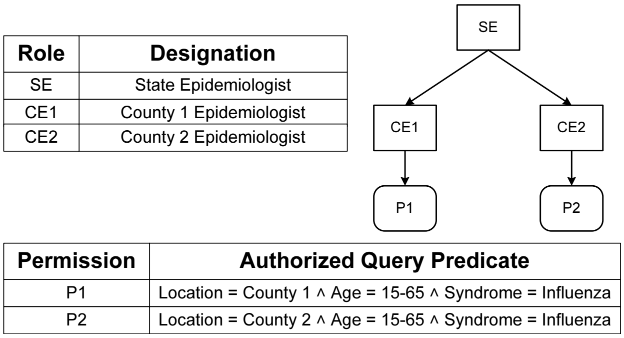
ORGANIZATIONS collect and analyze consumer data to improve services. Access Control Mechanisms (ACM) are used to ensure that only authorized information is available to users. However, sensitive information can still be misused by authorized insiders to compromise the privacy of consumers. The concept of privacy preservation for sensitive data can require the enforcement of privacy policies or the protection against identity disclosure by satisfying some privacy requirements [1]. In this paper, we investigate privacy preservation from the anonymity aspect. The sensitive information, even after the removal of identifying attributes, is still susceptible to linking attacks by the authorized insiders [2]. This problem has been studied extensively in the area of micro data publishing and privacy definitions, e.g., k-anonymity [2], l-diversity [3], t-closeness [4]. The anonymity techniques can be used with an access control mechanism to ensure both security and privacy of the sensitive information. However, privacy is achieved at the cost of accuracy and imprecision is introduced in the information provided by permissions under an access control policy. We use the concept of imprecision bound for each permission to set a threshold on the amount of imprecision that can be tolerated. To exemplify our approach, role-based access control is assumed. However, the approach is generic and can be applied to any security policy, e.g., discretionary access control or mandatory access control. The heuristics proposed in this paper for privacy-preserving access control are also relevant in the context of workload-aware anonymization.

The issue of protecting the privacy of individuals before releasing micro data containing personal information has been studied extensively during the last decade [5]. Anonymization algorithms use suppression and generalization of records to satisfy privacy requirements, e.g., k-anonymity and l-diversity with minimal distortion of micro data. Workload-aware anonymization techniques that minimize information loss for data mining tasks or for a given set of queries have been developed [6], [7]. However, the problem of satisfying accuracy constraints set by the multiple users of micro data has not been studied.

Example 1 (Motivating Scenario)**.** Syndromic surveillance systems are used at the state and federal levels to detect and monitor threats to public health [8], [9]. The department of health in a state collects the emergency department data (age, gender, location, time of arrival, symptoms, etc.) from county hospitals daily. Generally, each daily update consists of a static instance that is classified into syndrome categories by the department of health. Then, the surveillance data is anonymized and shared with departments of health at each county. An access control policy is given in Figure 1 that allows the roles to execute the authorized queries, e.g., Role CE1 can only execute queries under

- *Z. Pervaiz and A. Ghafoor are with Department of Electrical and Computer Engineering, W. Aref is with Computer Science Department and N. Prabhu is with Industrial Engineering Department, Purdue University, IN, 47907.*

Permission P1. The epidemiologists at the state and county level suggest community containment measures, e.g., isolation or quarantine according to the number of persons infected in case of a flu outbreak. According to the population density in a county, an epidemiologist can advise isolation if the number of persons reported with influenza are greater than 1000 and quarantine if that number is greater than 3000 in a single day. The anonymization adds imprecision to the query results and the imprecision bound for each query ensures that the results are within the tolerance required. If the imprecision bounds are not satisfied then unnecessary false alarms are generated due to the high rate of false positives.



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

Permission	Authorized Query Predicate
P1	Location = County 1 ^ Age = 15-65 ^ Syndrome = Influenza
P2	Location = County 2 ^ Age = 15-65 ^ Syndrome = Influenza

Fig. 1. Access control policy

From the perspective of access control, the motivation for the problem setting is that users are able to specify the error tolerance of each permission predicate for a given security policy or query workload. The anonymization techniques minimizing imprecision for all queries, e.g., Selection Mondrian [6], [10] give no clue about the imprecision that has been added to each permission/query in the anonymized micro data. Making the privacy requirement more stringent (e.g., increasing the value of k or l) results in more imprecision for queries. It is observed in the experiments in Section 5 that some permissions have more than 100% false positives. For some applications, when permissions have imprecision more than 50 to 100% of the size of the query, the access control module should either deny the request or give a warning.

The contributions of the paper are as follows. First, we introduce the concept of privacy-preserving access control. Second, we formulate the accuracy and privacy constraints as the problem of k-anonymous Partitioning with Imprecision Bounds (k-PIB) and give hardness results. Third, we propose heuristics to approximate the solution of the k-PIB 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 top-down heuristics for multidimensional partitioning to satisfy imprecision bounds. Experimental results are in Section 5, and in Section 6, an additional step to reduce the number of permissions violating imprecision bounds is proposed. The related work is presented in Section 7 and Section 8 concludes the paper.

2 Background

In this section, role-based access control concepts and privacy definitions based on anonymity are overviewed. Query evaluation semantics, imprecision, and the Selection Mondrian algorithm [10] are briefly explained.

Given a relation $T = \{A_1, A_2, \ldots, A_n\}$, where A_i is an attribute, T^* is the anonymized version of the relation T. The attributes can be of the following types:

- **Identifier.** Attributes, e.g., name and social security, that can uniquely identify an individual. These attributes are completely removed from the anonymized relation.
- **Quasi-Identifier (QI).** Attributes, e.g., gender, zip code, birth date, that can potentially identify an individual based on other information available to an adversary. QI attributes are generalized to satisfy the anonymity requirements.
- **Sensitive Attribute.** Attributes, e.g., disease or salary, that if associated to a unique individual will cause a privacy breach.

2.1 Role-based Access Control

Role-based Access Control (RBAC) allows defining permissions on objects based on roles in an organization. An RBAC policy configuration is composed of a set of Users (U), a set of Roles (R), and a set of Permissions (P). For the relational model, we assume that the set of permissions for a role are the selection predicates on the QI attributes that the role is authorized to execute [11]. Among the authorized tuple subset, a user is free to set any selection condition on the sensitive attribute. The user-to-role assignment (UA) is a user-to-role (U x R) mapping and the role-to-permission assignment (PA) is a role-to-permission (R x P) mapping. A role hierarchy (RH) defines an inheritance relationship among roles and is a partial order on roles (R x R) [12].

Definition 1 (RBAC Policy)**.** An RBAC policy ρ is a tuple $\langle U, R, P, UA, PA, RH \rangle$.

2.2 Anonymity Definitions

In this section, privacy definitions related to anonymity are introduced. The k-anonymity requirement is satisfied for an anonymized table if there are at least k tuples for every combination of predicates on the QI attributes [2].

Definition 2 (Equivalence Class (EC))**.** An equivalence class is a set of tuples having the same QI attribute values.

Definition 3 (k-anonymity Property)**.** A table T^* satisfies the k-anonymity property if each equivalence class has k or more tuples.

k-anonymity is prone to homogeneity attacks when the sensitive value for all the tuples in an equivalence class is the same. To counter this shortcoming l-diversity is introduced [3] and requires that each equivalence class of T^* must contain at least l distinct values of the sensitive attribute.

Definition 4 (l-diversity Principle)**.** A table T^* satisfies the l-diversity principle if each equivalence class contains at least l well-represented values of the sensitive attribute.

For sensitive numeric attributes, the l-diverse equivalence class can still leak information if the numeric values are close to each other. Variance diversity [6] and t-closeness [4] have been proposed for privacy protection against such a disclosure.

Definition 5 (Variance Diversity)**.** A table T^* is variance diverse if the variance V(EC) of each equivalence class satisfies $V(\text{EC}) \geq v$, where v is the variance diversity parameter.

	QI_1	QI_2	S_1
ID	Age	Zip	Disease
1	5	15	Flu
2	15	25	Fever
3	28	28	Diarrhea
4	25	15	Fever
5	22	28	Flu
6	32	35	Fever
7	38	32	Flu
8	35	25	Diarrhea

(a) Sensitive Table

	QI_1	QI_2	S_1
ID	Age	Zip	Disease
1	0-20	10-30	Flu
2	0-20	10-30	Fever
3	20-30	10-30	Diarrhea
4	20-30	10-30	Fever
5	20-30	10-30	Flu
6	30-40	20-40	Fever
7	30-40	20-40	Flu
8	30-40	20-40	Diarrhea

(b) 2-anonymous and 2-diverse Table

Fig. 2. Generalization for k-anonymity and l-diversity

The table in Figure 2(a) does not satisfy k-anonymity because knowing the age and zip code of a person allows associating a disease to that person. The table in Figure 2(b) is a 2-anonymous and 2-diverse version of table in Figure 2(a). The ID attribute is removed in the anonymized table and is shown only for identification of tuples. Here, for any combination of selection predicates on the zip code and age attributes, there are at least two tuples in each equivalence class. In Section 4, algorithms are presented for k-anonymity only. However, the experiments are performed for both l-diversity and variance diversity using the proposed heuristics for partitioning.

2.3 Predicate Evaluation and Imprecision

Various quality metrics have been proposed to evaluate and compare anonymization techniques, e.g., the *Discernibility Metric* (DM) [13], *Normalized average equivalence class size* [14] and the *Normalized Certainty Penalty* (NCP) metric [15]. These metrics do not capture the semantics of error bounds for predicate-based permissions. We first discuss the query predicate evaluation semantics and then the imprecision metric based on one possible predicate evaluation semantic. For query predicate evaluation over a table, say T, a tuple is included in the result if all the attribute values satisfy the query predicate. Here, we only consider conjunctive queries (The disjunctive queries can be expressed as a union of conjunctive queries), where each query can be expressed as a d-dimensional hyper-rectangle. The semantics for query evaluation on an anonymized table T^* needs to be defined. When the equivalence class partition (Each equivalence class can be represented as a d-dimensional hyper-rectangle) is fully enclosed inside the query region, all tuples in the equivalence class are part of the query result. Uncertainty in query evaluation arises when a partition overlaps the query region but is not fully enclosed. In this case, there can be many possible semantics. We discuss the following three choices: (i) Assuming uniform distribution of tuples in overlapping partitions, include tuples from all partitions according to the ratio of overlap between the query and the partition. Query evaluation under these semantics might under-count or over-count the query result depending upon the original distribution of tuples in the partition region. Most of the literature uses this uniform distribution semantic to compare anonymity techniques over selection tasks [14], [16]. However, the uniform distribution semantic is only valid for `COUNT` queries, as the choice of sensitive attribute value for the selected tuples from an overlapping partition is not defined. While, for access control, a tuple's QI attribute values along with the sensitive attribute value need to be returned. (ii) Include all tuples in all partitions that overlap the query region. This option will mostly add false positives to the original query result. (iii) Discard all tuples in all partitions that partially overlap the query region. This option will mostly have false negatives with respect to the original query result.

It is possible to define other query evaluation semantics. However, the error under any query evaluation scheme will reduce if the number of tuples in the partitions that overlap the query region can be minimized. For the rest of the paper, we focus on semantics under the second choice as defined in [10].

The imprecision quality metric defined in [10] is as follows:

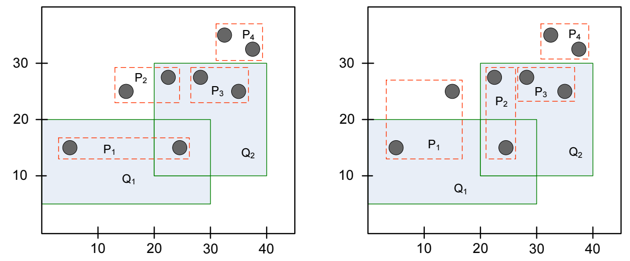
Definition 6 (Query Imprecision)**.** Query Imprecision is defined as the difference between the number of tuples returned by a query evaluated on an anonymized relation T^* and the number of tuples for the same query on the original relation T. The imprecision for query Q_i is denoted by imp_{Q_i},

$$imp_{Q_i} = |Q_i(T^*)| - |Q_i(T)| \text{ where} \tag{1}$$

$$|Q_i(T^*)| = \sum_{EC \text{ overlaps } Q_i} |EC|$$

The query Q_i is evaluated over T^* by including all the tuples in the equivalence classes that overlap the query region.

Example 2. The multi-dimensional partitioning for the table in Figure 2(a) satisfying 2-anonymity with minimal imprecision for queries Q_1 and Q_2 is given in Figure 3(a). The queries are the shaded rectangles with solid lines while the partitions are the regions enclosed by rectangles with dashed lines. The imprecision for Q_1 is 0, as Q_1's region overlaps only Partition P_1 with two tuples and the cardinality of Q_1 on original data is also two tuples. The imprecision for Query Q_2 is two because Q_2 overlaps partitions P_1, P_2, and P_3, $|Q_2(T^*)|$ = 6 while $|Q_2(T)|$ = 4.



(a) Anonymization with minimal imprecision

(b) Anonymization satisfying imprecision bounds

Fig. 3. Query evaluation over an anonymized table

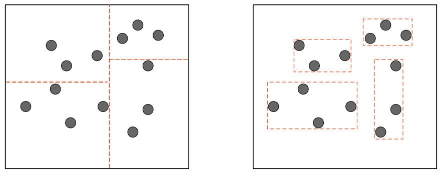
2.4 Recoding Techniques

Recoding techniques can be classified into global and local recoding techniques. Both the global and local recoding techniques have single and multi-dimensional variants. Our focus in this paper is on multi-dimensional recoding techniques as they have been shown to perform better than single-dimensional techniques [14]. In global recoding schemes, the tuple space is partitioned into non-overlapping hyper-rectangles and the anonymization algorithm maps all the tuples in a region to the same equivalence class. In contrast, the local recoding schemes partition the tuple space into overlapping hyper-rectangles. Anonymization algorithms based on local recoding can map multiple instances of the same tuple into different equivalence classes.

A taxonomy for recoding techniques has been presented in [17]. Global recoding is preferable if the anonymized data is to be used for classification or regression tasks [5] along with selection queries. However, for only selection queries local recoding techniques might perform better than global recoding [15], [17]. In this paper, only multi-dimensional global recoding is considered.

2.5 Top Down Selection Mondrian

Top Down Selection Mondrian (TDSM) algorithm is proposed by LeFevre et. al [10], [14] for a given query workload. This is the current state of the art for query-workload-based anonymization. The objective of TDSM is to minimize the total imprecision for all queries while the imprecision bounds for queries have not been considered. The anonymization for a given query workload with imprecision bounds has not investigated before to the best of our knowledge. We compare our results with TDSM in the experiments section. The algorithm presented in [14] is similar to the kd-tree construction [18]. TDSM starts with the whole tuple space as one partition and then partitions are recursively divided till the time new partitions meet the privacy requirement. To divide a partition, two decisions need to be made, i) Choosing a split value along each dimension, and ii) Choosing a dimension along which to split. In the TDSM algorithm [10], the split value is chosen along the median and then the dimension is selected along which the sum of imprecision for all queries is minimum. The time complexity of TDSM has not been reported in [10] and is $\mathcal{O}(d|Q|nlgn)$, where d is the number of dimensions of a tuple, Q is the set of queries, and n is the total number of tuples. The expression is derived by multiplying the height of the kd-tree with the work done at each level. The median cut generates a balanced tree with height lgn and the work done at each level is $d|Q|n$.



(a) Before compaction

(b) After compaction

Fig. 4. Compaction of partitions

2.5.1 *Compaction*

The partitions created by TDSM have dimensions along the median of the parent partition. A compaction procedure has been proposed in [7] where the created partitions are replaced by minimum bounding boxes. This step improves the precision of the anonymized table for any given query workload by reducing the overlapping partitions. An example of compaction is in Figure 4. In Section 5, compaction is carried for all the algorithms and then the results are compared.

3 Anonymization with Imprecision Bounds

In this section, we give definitions for the imprecision bound and the imprecision slack and formulate the problem of k-anonymous Partitioning with Imprecision Bounds (k-PIB).

3.1 Definitions

Let t_i be a tuple in Table T with d QI attributes. Tuple t_i can be expressed as a d-dimensional vector $\{v_1^{t_i}, \ldots, v_d^{t_i}\}$, where v_i is the value of the i^{th} attribute. Let D_{QI_i} be the domain of quasi-identifier attribute QI_i, then $t_i \in D_{QI_1} \times \ldots \times D_{QI_d}$. Any d-dimensional Partition P_i of the QI attribute domain space can be defined as a d-dimensional vector of closed intervals $\{I_1^{P_i}, \ldots, I_d^{P_i}\}$. The closed Interval $I_j^{P_i}$ is further defined as $[a_j^{P_i}, b_j^{P_i}]$, where $a_j^{P_i}$ is the start of the interval and $b_j^{P_i}$ is the end of the interval, and the length of the interval $l_j^{P_i}$ is $b_j^{P_i} - a_j^{P_i}$. A *multidimensional global recoding function*, e.g., Mondrian [14], first divides the d-dimensional QI attribute domain space into non-overlapping partitions $P_i \in P$, where each P_i is a d-dimensional rectangle. In the second step, the d-dimensional vector $\{v_1, \ldots, v_d\}$ for each tuple is replaced by the intervals $\{I_1^{P_i}, \ldots, I_d^{P_i}\}$ of the partition to which the tuple belongs. A Tuple, say t_j, belongs to a Partition, say P_l, if $\forall v_i^{t_j}, v_i^{t_j} \in I_i^{P_l} : a_i^{P_l} \leq v_i^{t_j} \leq b_i^{P_l}$.

Consider a set of queries Q, where $Q_i \in Q$ is defined by a boolean function of predicates on quasi-identifier attributes $\{QI_1, \ldots, QI_d\}$. A query defines a space in the domain of quasi-identifier attributes $D_{QI_1} \times \ldots \times D_{QI_d}$ and can be represented by a d-dimensional rectangle or a set of non-overlapping d-dimensional rectangles. To simplify the notation, we assume that Query Q_i is a single d-dimensional rectangle represented by $\{I_1^{Q_i}, \ldots, I_d^{Q_i}\}$. A Tuple t_j belongs to Query Q_i, if $\forall v_i^{t_j}, v_i^{t_j} \in I_i^{Q_i} : a_i^{Q_i} \leq v_i^{t_j} \leq b_i^{Q_i}$. Query Q_j and Partition P_l overlap if $\forall I_i^{Q_j} \forall I_i^{P_l}, a_i^{Q_j} \in I_i^{P_l}$ or $a_i^{P_l} \in I_i^{Q_j}$.

Definition 7 (Query Imprecision Bound)**.** The query imprecision bound, denoted by B_{Q_i}, is the total imprecision acceptable for a query predicate Q_i and is preset by the access control mechanism.

Example 3. For the table in Figure 2(a), assume that the imprecision bounds for Queries Q_1 and Q_2 are preset to 2 and 0. The partitioning given in Figure 3(a), although minimal, does not satisfy the imprecision bounds. However, the partitioning given in Figure 3(b) satisfies the bounds for Queries Q_1 and Q_2 as the imprecision for Q_1 and Q_2 is 2 and 0, respectively.

The specification of imprecision bounds is necessary because, from a user's perspective, all queries are not equal and the requirement of precision in some queries will be higher than others. This imprecision constraint sets an upper bound on the number of false positives returned for all authorized queries with the permitted query predicate.

Definition 8 (Query Imprecision Slack)**.** The query imprecision slack, denoted by s_{Q_i} for a Query, say Q_i, is defined as the difference between the query imprecision bound and the actual query imprecision.

$$s_{Q_i} = \begin{cases} B_{Q_i} - imp_{Q_i}, & \text{if } imp_{Q_i} \leq B_{Q_i} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Definition 9 (Partition Imprecision Cost (PIC))**.** The partition imprecision cost is a vector $\{ic_{P_i}^{Q_1}, \ldots, ic_{P_i}^{Q_n}\}$, where $ic_{P_i}^{Q_j}$ is the imprecision cost of a Partition $P_i \in P$ with respect to a Query Q_j. This cost is the number of tuples that are present in the partition but not in the query, i.e.,

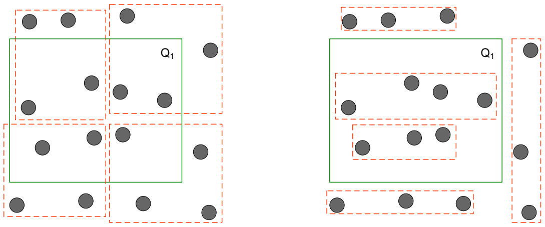
$$ic_{P_i}^{Q_j} = |P_i - Q_j| \tag{3}$$

where the minus sign denotes the set difference. The imprecision for a query imp_{Q_j}, defined in Equation 1, can also be expressed in terms of $ic_{P_i}^{Q_j}$ as

$$imp_{Q_j} = \mathop{\qquad}_{P_i \in P} ic_{P_i}^{Q_j}$$

The TDSM algorithm uses the median value along a dimension to split a partition. In the proposed heuristics in Section 4, query intervals are used to split the partitions that are defined as query cuts.

Definition 10 (Query Cut)**.** A query cut is defined as the splitting of a partition along the query interval values. For a query cut using Query Q_i, both the start of the query interval ($a_j^{Q_i}$) and the end of the query interval ($b_j^{Q_i}$) are considered to split a partition along the j^{th} dimension.



(a) Median cut

(b) Query cut

Fig. 5. Comparison of median and query Cut

Example 4. A comparison of median cut and query cut is given in Figure 5 for 3-anonymity. The rectangle with solid lines represents Query Q_1. While, the rectangles with dotted lines represent partitions. In Figure 5(a) the tuples are partitioned according to the median cut and even after dividing the tuple space into four partitions there is no reduction in imprecision for the Query Q_1. However, for query cuts in Figure 5(b) the imprecision is reduced to zero as partitions are either non-overlapping or fully enclosed inside the query region.

3.2 The k-PIB Problem

The optimal k-anonymity problem has been shown to be NP-complete for suppression [19] and generalization [20]. It has also been proved that the optimal k-anonymity is strong-sense NP-hard [21]. The hardness result for k-PIB follows the construction of Lefevre et al. [14] that shows the hardness of k-anonymous multi-dimensional partitioning with the smallest average equivalence class size. We show that finding k-anonymous partitioning that satisfies imprecision bounds for minimum number of queries is also NP-hard. A multiset of tuples is transformed into an equivalent set of distinct $(tuple, count)$ pairs. The cardinality of Query Q_i is the sum of count values of tuples falling inside the query hyper-rectangle. The constant qv defines an upper bound for the number of queries that can violate the bounds. The decision version of the k-PIB problem is as follows:

Definition 11 (Decisional k-anonymity with Imprecision Bounds)**.** Given a set $t \in T$ of unique $(tuple, count)$ pairs with tuples in the d-dimensional space and a set of queries $Q_i \in Q$ with imprecision bounds B_{Q_i}, does there exist a multidimensional partitioning for T such that for every resulting multidimensional region R_i, $\sum_{t \in R_i} count(t) \geq k$ or $\sum_{t \in R_i} count(t) = 0$, and number of queries having $imp_{Q_i} > B_{Q_i}$ is less than the positive constant qv?

Theorem 3.1. *Decisional k-anonymity with Imprecision Bounds is NP-complete.*

Proof: Refer to Appendix A. □

3.3 Privacy-preserving Access Control

A privacy-preserving access control framework, illustrated in Figure 6, is proposed where the privacy protection mechanism ensures that the privacy and accuracy goals are met before the sensitive data is available to the access control mechanism. The access control policies define permissions for roles based on selection predicates. Privacy Protection Mechanisms (PPM) use suppression and generalization to anonymize and satisfy privacy requirements. The attainment of the privacy goals is achieved at the cost of the precision of the data available to the authorized users. The access control mechanism needs to specify the level of imprecision that can be tolerated by the user for each permission. This specification of the imprecision bound ensures that the authorized information has the desired level of accuracy. Then, the privacy protection mechanism needs to meet the privacy requirement along with the imprecision bound for each permission.

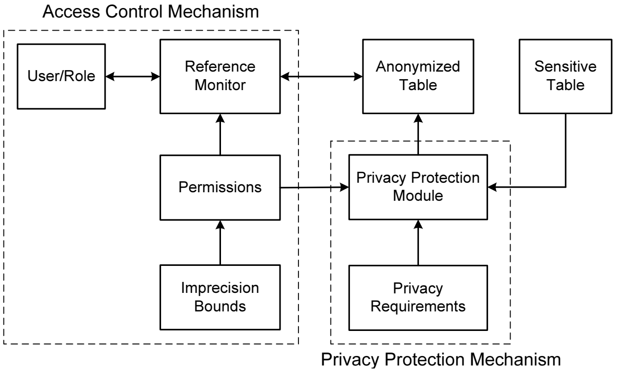


Fig. 6. Privacy-preserving access control

Definition 12 (Privacy-preserving RBAC Policy)**.** A privacy-preserving RBAC policy ρ_p is a tuple $\langle U, R, P, B_p, UA, PA, RH \rangle$, where B_p is the imprecision bound for the allowed query predicate Q_i under permission $p \in P$.

Notice that we use permission and query interchangeably.

3.4 Expected Query Imprecision

Given n tuples, it is assumed that the tuples are uniformly distributed in the domain space of the QI attributes. In order to estimate the expected imprecision for a randomly selected query, first the expected number of partitions overlapping the query needs to be found. We use the approach by Otoo et. al [22], where they find overlapping intervals in each dimension and then take the product to get the expected number of overlapping partitions. However, we still need to find the expected partition size $|P_e|$ and expected length of intervals $l_i^{P_e}$. We use the domain length of each attribute in domain space and then divide this length of first QI attribute by 2. The length of interval $l_1^{P_e}$ is updated and the new partition will now contain $\frac{n}{2}$ tuples. For the next division, another QI attribute is selected and the process is repeated until the expected partition size is $k \leq |P_e| < 2k$.

Lemma 3.2. *The expected imprecision for a query Q_j is*

$$E(imp_{Q_j}) = \left(\prod_{i=1}^{d} \left\lfloor \frac{l_i^{Q_j} + l_i^{P_e} - 1}{l_i^{P_e}} \right\rfloor \right) * |P_e| - |Q_j| \quad (4)$$

In this equation, we round up the fraction ($l_i^{Q_j}$ divided by $l_i^{P_e}$) and then take the floor in each dimension. Multiplying the number of partitions with the expected size of each partition gives the expected number of tuples in the query $|Q_j(T^*)|$. Subtracting the original size $|Q_j|$ of the query gives the expected imprecision.

Example 5. Consider a query with range 10-21 and 5-10 for two attributes and a query size of 50. If the expected partition length for the two attributes is 3 and 2 and the expected partition size is 6 then 12 partitions are expected to overlap the query. The expected query imprecision will be 22 (12*6 - 50) tuples.

4 Heuristics for Partitioning

In this section, three algorithms based on greedy heuristics are proposed. All three algorithms are based on kd-tree construction [18]. Starting with the whole tuple space the nodes in the kd-tree are recursively divided till the partition size is between k and $2k$. The leaf nodes of the kd-tree are the output partitions that are mapped to equivalence classes in the given table. Heuristic 1 and 2 have time complexity of $\mathcal{O}(d|Q|^2n^2)$. Heuristic 3 is a modification over Heuristic 2 to have $\mathcal{O}(d|Q|nlgn)$ complexity, which is same as that of TDSM. The proposed query cut can also be used to split partitions using bottom-up (R$^+$-tree) techniques [7].

4.1 Top-Down Heuristic 1 (TDH1)

In TDSM, the partitions are split along the median. Consider a partition that overlaps a query. If the median also falls inside the query then even after splitting the partition, the imprecision for that query will not change as both the new partitions still overlap the query as illustrated in Figure 5. In this heuristic, we propose to split the partition along the query cut and then choose the dimension along which the imprecision is minimum for all queries. If multiple queries overlap a partition, then the query to be used for the cut needs to be selected. The queries having imprecision greater than zero for the partition are sorted based on the imprecision bound and the query with minimum imprecision bound is selected. The intuition behind this decision is that the queries with smaller bounds have lower tolerance for error and such a partition split ensures the decrease in imprecision for the query with the smallest imprecision bound. If no feasible cut satisfying the privacy requirement is found, then the next query in the sorted list is used to check for partition split. If none of the queries allow partition split, then that partition is split along the median and the resulting partitions are added to the output after compaction.

The TDH1 algorithm is listed in Algorithm 1. In the first line, the whole tuple space is added to the set of candidate partitions. In the Lines 3-4, the query overlapping the candidate partition with least imprecision bound and imprecision greater than zero is selected. The **while** loop in Lines 5-8 checks for a feasible split of the partition along query intervals. If a feasible cut is found, then the resulting partitions are added to CP. Otherwise, the candidate partition is checked for median cut in Line 12. A feasible cut means that each partition resulting from split should satisfy the privacy requirement. The traversal of the kd-tree for partitions to consider in Set CP can be depth-first or breadth-first. However, the order of traversal for TDH1 does not matter.

This heuristic of selecting cuts along minimum bound queries favors queries with smaller bounds. This behavior is also evident in the experiments in Section 5 for the randomly selected query workload. However, this approach creates imprecision slack in the queries with smaller bounds that could have been used to satisfy bounds of other queries.

Algorithm 1: TDH1

```
   Input  : $T$, $k$, $Q$, and $B_{Q_j}$
   Output: $P$
1  Initialize Set of Candidate Partitions($CP \leftarrow T$)
2  for ($CP_i \in CP$) do
3      Find the set of queries $QO$ that overlap $CP_i$
       such that $ic^{QO_j}_{CP_i} > 0$
4      Sort queries $QO$ in increasing order of $B_{Q_j}$
5      while (feasible cut is not found) do
6          Select query from $QO$
7          Create query cuts in each dimension
8          Select dimension and cut having least
           overall imprecision for all queries in $Q$
9      if (Feasible cut found) then
10         Create new partitions and add to $CP$
11     else
12         Split $CP_i$ recursively along median till
           anonymity requirement is satisfied
13         Compact new partitions and add to $P$
14 return ($P$)
```

Lemma 4.1. *The time complexity of TDH1 is $\mathcal{O}(d|Q|^2n^2)$.*

Proof: The time complexity is derived by multiplying the height of the kd-tree with the work performed at each level. The height of the kd-tree for TDH1 in the worst case can be $\frac{n}{k}$, which occurs when each successive cut creates one partition of exactly size k. In the worst case, at each level we might have to check all queries for a feasible cut, which leads to $d|Q|^2n$. The total time complexity is then $\mathcal{O}(d|Q|^2n^2)$. □

4.2 Top-Down Heuristic 2 (TDH2)

In the Top-Down Heuristic 2 algorithm (TDH2, for short), the query bounds are updated as the partitions are added to the output. This update is carried out by subtracting the $ic^{Q_j}_{P_i}$ value from the imprecision bound B_{Q_j} of each query, for a Partition, say P_i, that is being added to the output. For example, if a partition of size k has imprecision 5 and 10 for Queries Q1 and Q2 with imprecision bound 100 and 200, then the bounds are changed to 95 and 190, respectively. The best results are achieved if the kd-tree traversal is depth-first (preorder). Preorder traversal for the kd-tree ensures that a given partition is recursively split till the leaf node is reached. Then, the query bounds are updated. Initially, this approach favors queries with smaller bounds. As more partitions are added to the output, all the queries are treated fairly. During the query bound update, if the imprecision bound for any query gets violated, then that query is put on low priority by replacing the query bound by the query size. The intuition behind this decision is that whatever future partition splits TDH2 makes, the query bound for this query cannot be satisfied. Hence, the focus should be on the remaining queries.

Algorithm 2: TDH2

```
   Input  : $T$, $k$, $Q$, and $B_{Q_j}$
   Output: $P$
1  Initialize Set of Candidate Partitions($CP \leftarrow T$)
2  for ($CP_i \in CP$) do
       // Depth-first(preorder) traversal
3      Find the set of queries $QO$ that overlap $CP_i$
       such that $ic_{CP_i}^{QO_j} > 0$
4      Sort queries $QO$ in increasing order of $B_{Q_j}$
5      while (feasible cut is not found) do
6          Select query from $QO$
7          Create query cuts in each dimension
8          Select dimension and cut having least
           overall imprecision for all queries in $Q$
9      if (Feasible cut found) then
10         Create new partitions and add to $CP$
11     else
12         Split $CP_i$ recursively along median till
           anonymity requirement is satisfied
13         Compact new partitions and add to $P$
14         Update $B_{Q_j}$ according to $ic_{P_i}^{Q_j}$, $\forall Q_j \in Q$
15 return ($P$)
```

The algorithm for TDH2 is listed in Algorithm 2. There are two differences compared to TDH1. First, the kd-tree traversal for the **for** loop in Lines 2-14 is preorder. Second, in Line 14, the query bounds are updated as the partitions are being added to the output (P). The time complexity of TDH2 is $\mathcal{O}(d|Q|^2n^2)$, which is the same as that of TDH1. In Section 4.3, we propose changes to TDH2 that reduce the time complexity at the cost of increased query imprecision.

4.3 Top-Down Heuristic 3 (TDH3)

The time complexity of the TDH2 algorithm is $\mathcal{O}(d|Q|^2n^2)$, which is not scalable for large datasets (greater than 10 million tuples). In the Top-Down Heuristic 3 algorithm (TDH3, for short), we modify TDH2 so that the time complexity of $\mathcal{O}(d|Q|nlgn)$ can be achieved at the cost of reduced precision in the query results. Given a partition, TDH3 checks the query cuts only for the query having the lowest imprecision bound. Also, the second constraint is that the query cuts are feasible only in the case when the size ratio of the resulting partitions is not highly skewed. We use a skew ratio of 1:99 for TDH3 as a threshold. If a query cut results in one partition having a size greater than hundred times the other, then that cut is ignored. TDH3 algorithm is listed in Algorithm 3. In Line 4 of Algorithm 3, we use only one query for the candidate cut. In Line 6, the partition size ratio condition needs to be satisfied for a feasible cut. If a feasible query cut is not found, then the partition is split along the median as in Line 11.

Algorithm 3: TDH3

```
   Input  : $T$, $k$, $Q$, and $B_{Q_j}$
   Output: $P$
1  Initialize Set of Candidate Partitions($CP \leftarrow T$)
2  for ($CP_i \in CP$) do
       // Depth-first(preorder) traversal
3      Find the set of queries $QO$ that overlap $CP_i$
       such that $ic_{CP_i}^{QO_j} > 0$
4      Select query from $QO$ with smallest $B_{Q_j}$
5      Create query cuts in each dimension
6      Reject cuts with skewed partitions
7      Select dimension and cut having least overall
       imprecision for all queries in $Q$
8      if (Feasible cut found) then
9          Create new partitions and add to $CP$
10     else
11         Split $CP_i$ recursively along median till
           anonymity requirement is satisfied
12         Compact new partitions and add to $P$
13         Update $B_{Q_j}$ according to $ic_{P_i}^{Q_j}$, $\forall Q_j \in Q$
14 return ($P$)
```

Lemma 4.2. *The time complexity TDH3 is $\mathcal{O}(d|Q|nlgn)$.*

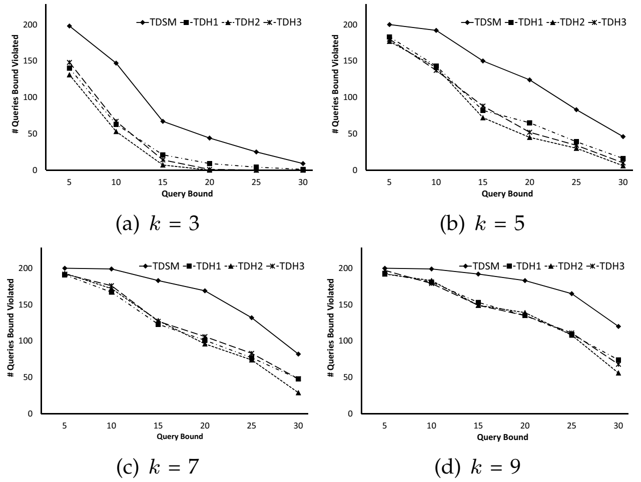
Proof: The height of the kd-tree for TDH3 will be $log_{\frac{100}{99}}n$. The work performed at each level of the kd-tree is $|Q|n$ as only one query is considered for a feasible cut. This gives a total time complexity of $\mathcal{O}(d|Q|nlgn)$. □

The time complexity of TDH3 is $\mathcal{O}(d|Q|nlgn)$ with a constant factor of $log_{\frac{100}{99}}$ in comparison to TDSM.

5 Experiments

The experiments have been carried out on two datasets for the empirical evaluation of the proposed heuristics. The first dataset is the Adult dataset from the UC Irvine Machine Learning Repository [23] having 45222 tuples and is the de facto benchmark for k-anonymity research. The attributes in the Adult dataset are: Age, Work class, Education, Marital status, Occupation, Race, and, Gender. The second dataset is the Census dataset [24] from IPUMS[1]. This dataset is extracted for Year 2001 using attributes: Age, Gender, Marital status, Race, Birth place, Language, Occupation, and Income. The size of the dataset is about 1.2 million tuples. For the k-anonymity experiments, we use the first eight attributes as the QI attributes. For the l-diversity experiments, we use Attribute *occupation* as the sensitive attribute and the first seven attributes as the QI attributes. For the l-diversity experiments, all the tuples having the occupation value as Not Applicable (0 in the dataset) are removed, which leaves about 700k tuples. In the case of the variance diversity experiments, Attribute *income* is used as the sensitive attribute and all the tuples having the income value as Not Applicable (9999999 in the dataset) are removed, which leaves about 950k tuples.

We use 200 and 500 queries generated randomly as the workload/permissions for the Adult dataset and Census dataset, respectively. The experiments have been conducted for two types of query workloads. To avoid yielding too many empty queries, the queries are generated randomly using the approach by Iwuchukwu, et al. [7]. 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. To simulate the permissions for an access control policy, the query selectivity for both the datasets is set to range from 0.5% to 5%. For the first workload, if the query output is between 500 to 5500 tuples for the Adult dataset and 1000 to 50,000 for the Census dataset, the query is added to the workload. For the second workload (we will refer to this workload as the uniform query workload) this range (1000 to 50,000 for Census dataset) is divided into ten equal intervals and we add only 50 queries from each interval to the workload. Similarly, for the Adult dataset, 20 queries are added from each size interval. The first workload is used for the l-diversity and variance diversity experiments. The average query size for the Adult dataset is 3000 and for the Census dataset is 25,000 for the uniform query workload. The imprecision bounds for all queries are set based on the query size for the current experiment. Otherwise, bounds for queries can be set according to the precision required by the access control mechanism. The intuition behind setting bounds as a factor of the query size is that imprecision added to the query is proportional to the query size.



(a) k = 3

(b) k = 5

(c) k = 7

(d) k = 9

Fig. 7. No of queries whose bounds are not satisfied for k-anonymity(Adult dataset)

For the k-anonymity experiments, we fix the value of k and change the query imprecision bounds from 5% to 30% with increments of 5. Then, we find the number of queries whose bounds have not been satisfied by each algorithm for the uniform query workload. The results for k-anonymity are given in Figure 7 for the Adult dataset for k values of 3, 5, 7 and 9. Heuristic TDH2 has the least number of query bound violations and is better than TDH1 because of TDH2's query-bound update step. TDH3 with added constraints and reduced complexity also performs better than TDSM. The number of queries not satisfying imprecision bounds increases as the value of k increases. The focus is to maximize the number of queries satisfying imprecision bounds even if the total imprecision as compared to TDSM is increased. However, as in Figure 8, even the total imprecision for all the proposed heuristics is considerably less than TDSM for all values of k. Due to limited space, only the above results are discussed for the Adult dataset.

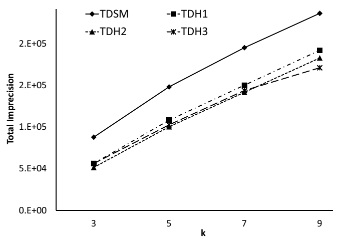
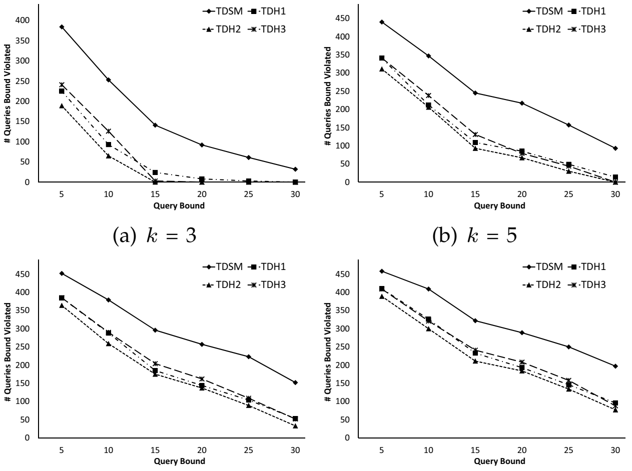


Fig. 8. Total imprecision for all queries(Adult dataset)

For k-anonymity, the number of queries for which the imprecision bound is not satisfied is given in Figure 9 for the Census dataset using the uniform query workload of 500 queries. The results have the same behavior as that for the Adult dataset. In both cases, TDH2 has the lowest number of queries violat-

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



(a) k = 3

(b) k = 5

(c) k = 7

(d) k = 9

Fig. 9. No of queries whose bounds are not satisfied for k-anonymity(Census dataset)

ing the imprecision bounds. The sum of imprecision for all queries is given in Figure 10, where TDH2 also has the lowest total imprecision for all values of k. In Figure 9, the total number of violated queries is given. So, in Figure 11, we plot the number of queries against the margin by which they violate the query bound (Imprecision bound is set as 25% of the query size). Six query imprecision ranges have been considered that are: imprecision is less than 10%, 10-25%, 25-50%, 50-75%, 75-100% and greater than 100% of the bound. In Section 6, an algorithm is proposed to realign the output partitions to satisfy the imprecision bounds of queries that violate the bound by a less than 10% margin. The reason for using the uniform query workload (50 randomly selected queries from each size range having cardinality between 0.5% to 5% of the dataset) is that it helps observe the behavior of the queries violating the bounds for each algorithm. Intuitively, there is more chance of violating the imprecision bounds for a query having a smaller imprecision bound. In Figure 12, the number of queries violated for each size range (10 size intervals in 1k-50k) are plotted. The behavior of TDSM follows the intuition as more queries in the smaller size range are violated. For TDH1, the heuristic always favors the queries with smaller bounds when being considered for a partition split. Thus, for TDH1, less queries are violated of smaller bounds than of larger ones[2]. TDH2 and TDH3 favor queries with smaller bounds initially. However, as partitions are added to the output, all queries are treated fairly. Hence, the number of queries violated is almost uniform in this case.

We use the same heuristics for the privacy requirements of l-diversity and variance diversity. The experiments are conducted for l values of 7 and 9. For each value of l, we change the query imprecision bounds from 5% to 30% with increments of 5 and find the number of queries whose bounds are not satisfied by each algorithm. The results for l values of 7 and 9 are given in Figure 13. The results show that TDH2 violates the bound for a less number of queries for l-diversity.

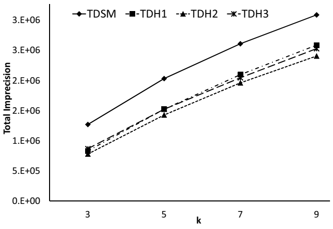
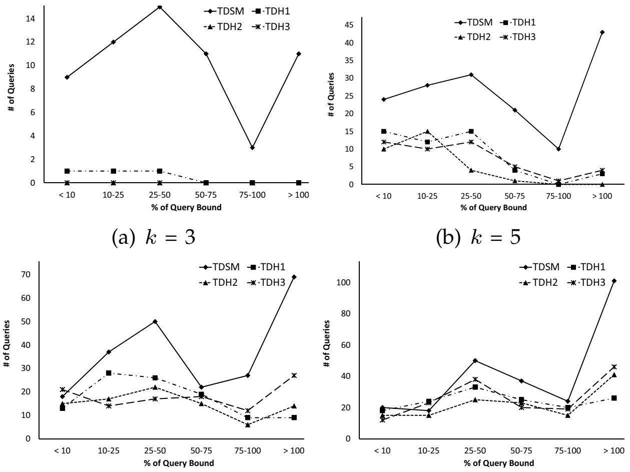


Fig. 10. Total imprecision for all queries(Census dataset)



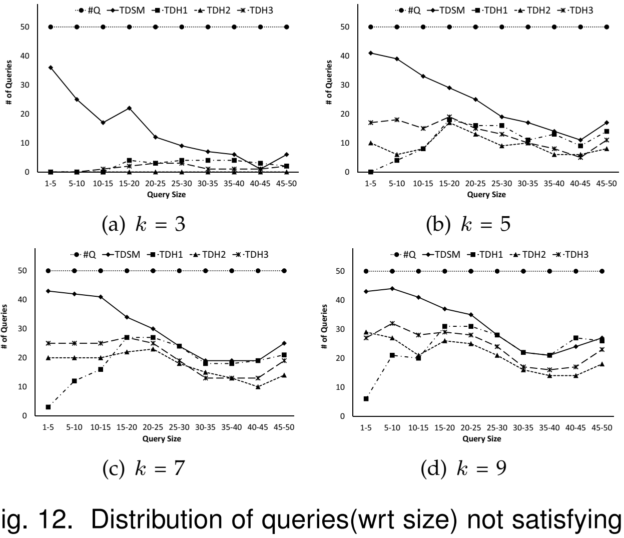
(a) k = 3

(b) k = 5

(c) k = 7

(d) k = 9

Fig. 11. Distribution of queries(wrt bound) not satisfying bound at 25% for k-anonymity(Census dataset)



(a) k = 3

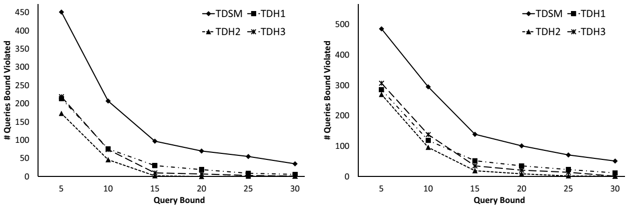
(b) k = 5

(c) k = 7

(d) k = 9

Fig. 12. Distribution of queries(wrt size) not satisfying bound at 15% for k-anonymity(Census dataset)

[2] We are using size and bound interchangeably as an imprecision bound is a fraction of the query size.

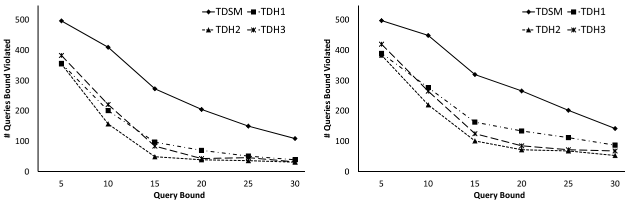


(a) l = 5

(b) l = 7

Fig. 13. Number of queries not satisfying bound for l-diversity(Census dataset)

In the case of variance diversity 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 variance diversity value , we change the query imprecision bounds from 5% to 30% and find the number of queries whose bounds are violated by each algorithm. The results for variance diversity are given in Figure 14. For variance diversity, TDH2 gives the best results.



(a) $v = \frac{V}{200}$

(b) $v = \frac{V}{100}$

Fig. 14. Number of queries not satisfying bound for variance-diversity(Census dataset)

In the next experiment, all the algorithms are compared with respect to the size of the given query set. The size of the query set is changed from 32 to 1024 for a k value of 5 and a query imprecision bound of 30%. Observe in Figure 15 that as the size of query workload is increased bounds for more queries are violated. However, the proposed heuristics still violate bounds of less queries than TDSM.

While the intention is to satisfy the imprecision bounds for as many queries as possible from the given set of queries, it is as important to maintain the utility of all other queries. In this experiment, after partitioning for a given set of queries, we generate 1000 new random queries and compare the number of queries satisfied at 30% imprecision bound by each algorithm. The results are given in Figure 16. Observe that the performance of all the algorithms is similar. The slightly better results in case of TDH1, TDH2, and TDH3 are due to the fact that more queries are picked from high density tuple regions for which partitioning is already optimized for the proposed heuristics.

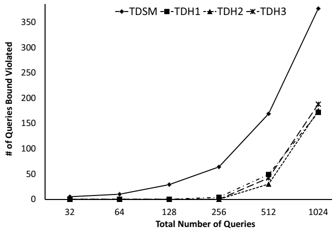


Fig. 15. Varying size of given query workload(Census dataset)

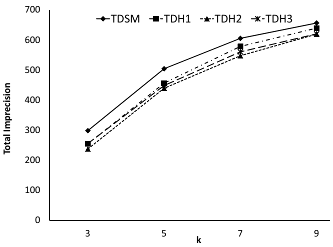


Fig. 16. Performance for a different query workload (Census dataset)

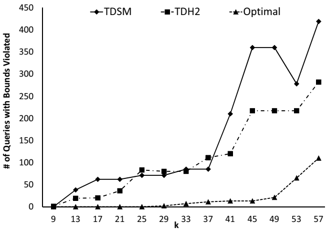
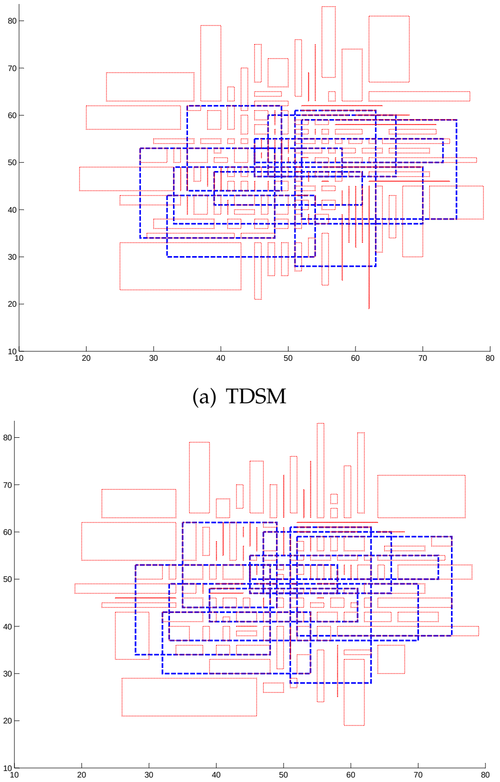


Fig. 17. Comparison with optimal solution

The proposed techniques do not provide any performance guarantees. However, we compare the performance of the proposed heuristics with the optimal solution using a smaller subset of the Adult dataset. We use three attributes (Work Class, Marital Status, and, Race) and pick 1000 tuples randomly from the Adult dataset. The heuristic algorithms are executed using a workload of 1000 randomly selected queries with an imprecision bound of 20% of the size of query. For the optimal partitioning, all possible partitions are created based on the selected three attributes. In the next step, the partitions having less than k tuples or more than $2d(k-1)+f_{max}$ [14] are rejected, where f_{max} is the maximum frequency of any tuple in the partition. For the remaining partitions, an integer programming model in GAMS [25] is executed to select a set of partitions containing all the tuples while violating the imprecision bound for the minimum number of queries. The comparison of the optimal partitioning for the least number of query imprecision bound violations against TDSM and TDH2 is given in Figure 17. Observe that as the value of k is increased, the gap between TDH2 and the optimal solution increases suggesting that the quality factor is dependent on k.

The visual representation of the partitions resulting from the proposed heuristic TDH2 and TDSM is given in Figure 18. Here, 1000 tuples with two attributes



(a) TDSM

(b) TDH2

Fig. 18. Anonymization for two attributes with discrete normal distribution ($\mu = 50, \sigma = 10$)

are randomly selected (Normal distribution with μ = 50, σ = 10, and cardinality = 100). 10 random queries are also selected (Query selectivity is from 10% to 50%) and the query imprecision bound is set to 10% of the query size. The rectangles with the blue (darker) lines are the queries while the rectangles with red (lighter) lines are partitions generated by the heuristics at k = 5. The query imprecision results are given in Table 1. Observe that in Figure 18, less partitions are overlapping the query region for TDH2 as compared to TDSM, e.g., Query Q2 (range: 32-54, 30-43) has zero imprecision under TDH2 and all the partitions are fully enclosed by the query region.

TABLE 1
Query Imprecision

Query	Dimensions	Size	Bound	Query Imprecision	
				TDSM	TDH2
Q1	45-58, 47-53	148	14	43	0
Q2	32-54, 30-43	152	15	27	0
Q3	45-73, 50-55	187	18	2	8
Q4	39-61, 41-48	194	19	34	0
Q5	35-49, 44-62	260	26	25	18
Q6	28-48, 34-53	271	27	37	8
Q7	47-66, 47-60	296	29	52	21
Q8	52-75, 38-59	320	32	57	17
Q9	51-63, 28-61	353	35	36	22
Q10	33-70, 37-49	363	36	41	21

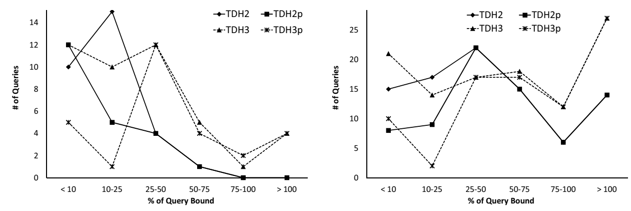
6 Improving the Number of Queries Satisfying the Imprecision Bounds

In Section 3, the query imprecision slack is defined as the difference between the query bound and query imprecision. This query imprecision slack can help satisfy queries that violate the bounds by only a small margin by increasing the imprecision of the queries having more slack. The margin by which queries violate the bounds is given in Figure 11. In this repartitioning step, we consider only the first two groups of queries that fall within 10% and 10-25% of the bound only and these queries are added to the Candidate Query set (CQ), while all queries satisfying the bounds are added to the query set QS. The output partitions are all the leaf nodes in the kd-tree. For repartitioning, we only consider those pairs of partitions from the output that are siblings in the kd-tree and have imprecision greater than zero for the queries in the candidate query set. These pairs of partitions are then added to the candidate partition set for repartitioning. Merging such a pair of sibling leaf nodes ensures that we still get a hyper-rectangle and the merged partition is non-overlapping with any other output partition. The repartitioning is first performed for the set of queries within 10% of the bound. The partitions that are modified are removed from the candidate set and then the second group of queries is checked. The algorithm for repartitioning is listed as Algorithm 4. In Lines 6-9, we check if a query cut along any dimension exists that reduces the total imprecision for the queries in CQ Set while still satisfying the bounds of the queries in QS. If such a cut exists, then the old partitions are removed and the new ones are added to Output P in Lines 11-12. After every iteration, the imprecision of the queries in Set CQ is checked. If the imprecision is less than the bound for any query, then as in Line 15, that query is moved from Set CQ to QS. The proposed algorithm in the experiments satisfies most of the queries from the first group and only a few queries from the second group. This repartitioning step is equivalent to partitioning all the leaf nodes that in the worst case can take $\mathcal{O}(|Q|n)$ time for each candidate query set.

In the experiments, we set the value of k to 5 and 7 with a query imprecision bound of 30% of the query size. The results for repartitioning are given in Figure 19. TDH2p and TDH3p are the results after the repartitioning step. Observe that most of the queries in the 10% group have been satisfied, while for the 10-25% group, some of these have been satisfied while the others have moved into the first group.

Algorithm 4: Post Processing

```
Input : $T$, $k$, $Q$, $P$, and $B_q$
Output: $P$
Initialize $SQ$, $CQ$, and $CP$
Add $q \in Q$ satisfying bound to $SQ$
Add $q \in Q$ violating bound by 10% to Candidate Query set($CQ$)
Add all sibling leaf node pairs having $\sum_{q \in CQ}(ic^{q_j}_{P_i} + ic^{q_j}_{P_{i+1}}) > 0$ to Candidate Partition($CP$)
for ($CP_i \in CP$) do
    Merge the first pair $CP_i$ and $CP_{i+1}$
    Select $q$ from $CQ$ with the least imprecision greater than the imprecision bound
    Create the candidate cuts in each dimension
    Select the cut and the dimension satisfying all $q \in SQ$ with the minimum imprecision $\forall q \in CQ$
    if (feasible cut found) then
        Remove $CP_i$ and $CP_{i+1}$ from $CP$ and $P$
        Add new partitions to $P$
        for ($q \in CQ$) do
            if ($Imp_q < B_q$) then
                Remove $q$ from $CQ$ and add to $SQ$
return ($P$)
```



(a) k = 5 and Bound = 30%

(b) k = 7 and Bound = 30%

Fig. 19. Improvements after repartitioning for k-anonymity(Census dataset)

Repartitioning of the other groups of queries reduces the total imprecision but the gains in terms of having more queries satisfying bounds are not worthwhile.

7 Related Work

Access control mechanisms for databases allow queries only on the authorized part of the database [26], [27]. Predicate-based fine-grained access control has further been proposed, where user authorization is limited to pre-defined predicates [11]. Enforcement of access control and privacy policies have been studied in [28]. However, studying the interaction between the access control mechanisms and the privacy protection mechanisms has been missing. Recently, Chaudhuri et al. have studied access control with privacy mechanisms [29]. They use the definition of *differential privacy* [30] whereby random noise is added to original query results to satisfy privacy constraints. However, they have not considered the accuracy constraints for permissions. We define the privacy requirement in terms of k-anonymity. It has been shown by Li et. al [31] that after sampling, k-anonymity offers similar privacy guarantees as those of *differential privacy*. From an access control user perspective, the permissions based on selection predicates have different accuracy requirements that need to be satisfied by the privacy protection mechanism. The proposed privacy-aware access control framework allows the access control mechanism to specify imprecision constraints that the privacy protection mechanism is required to meet along with the privacy requirements.

The challenges of privacy-aware access control are similar to the problem of workload-aware anonymization. In our analysis of the related work, we focus on query-aware anonymization. For the state of the art in k-anonymity techniques and algorithms, we refer the reader to the recent survey papers [5], [32]. Workload-aware anonymization is first studied by Lefevre et al. [6], [10]. They have proposed the Selection Mondrian algorithm, which is a modification to the greedy multidimensional partitioning algorithm Mondrian [14]. In their algorithm, based on the given query-workload, the greedy splitting heuristic minimizes the sum of imprecision for all queries. Iwuchukwu et al. have proposed an R$^+$-tree [33] based anonymization algorithm [7], [16]. The authors illustrate by experiments that anonymized data using biased R$^+$-tree based on the given query workload is more accurate for those queries than for an unbiased algorithm. Ghinita et al. have proposed algorithms based on space filling curves for k-anonymity and l-diversity [34]. They also introduce the problem of accuracy-constrained anonymization for a given bound of acceptable information loss for each equivalence class [35]. Similarly, Xiao et. al [36] propose to add noise to queries according to the size of the queries in a given workload to satisfy differential privacy. However, bounds for query imprecision have not been considered. The existing literature on workload-aware anonymization has a focus to minimize the overall imprecision for a given set of queries. However, anonymization with imprecision constraints for individual queries has not been studied before. We follow the imprecision definition of Lefevre et al. [6] and introduce the constraint of imprecision bound for each query in a given query workload.

8 Conclusions

A privacy-preserving access control framework assuming a relational model has been proposed. The framework is a combination of access control and privacy protection mechanisms. The access control mechanism allows only authorized query predicates on sensitive data. The privacy preserving module anonymizes the data to meet privacy requirements and imprecision constraints on predicates set by the access control mechanism. We formulate this interaction as the problem of k-anonymous Partitioning with Imprecision Bounds(k-PIB). We give hardness results for the k-PIB problem and present heuristics for partitioning the data to the satisfy the privacy constraints and the imprecision bounds. In the current work, static access control and relational data model has been assumed. For future work, we plan to extend the proposed privacy-preserving access control to incremental data and dynamic access control.

Acknowledgments

The research in this paper was partially supported by the National Science Foundation under Grants IIS-1117766, IIS-0964639, and IIS-0811954.

References

[1] E. Bertino and R. Sandhu, "Database security-concepts, approaches, and challenges," *Dependable and Secure Computing, IEEE Transactions on*, vol. 2, no. 1, pp. 2–19, 2005.

[2] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1010–1027, 2001.

[3] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[4] N. Li, T. Li, and S. Venkatasubramanian, "Closeness: A new privacy measure for data publishing," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 7, pp. 943–956, 2010.

[5] B. Fung, K. Wang, R. Chen, and P. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[6] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Workload-aware anonymization techniques for large-scale datasets," *ACM Transactions on Database Systems (TODS)*, vol. 33, no. 3, pp. 1–47, 2008.

[7] T. Iwuchukwu, *Anonymization techniques for large and dynamic data sets*. PhD thesis, The University of Wisconsin-Madison, 2008.

[8] 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," *Advances in Disease Surveillance*, vol. 6, no. 3, pp. 1–20, 2008.

[9] S. Grannis, M. Wade, J. Gibson, and J. Overhage, "The indiana public health emergency surveillance system: Ongoing progress, early findings, and future directions," in *AMIA Annual Symposium proceedings*, vol. 2006, p. 304, American Medical Informatics Association, 2006.

[10] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Workload-aware anonymization," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 277–286, ACM, 2006.

[11] S. Chaudhuri, T. Dutta, and S. Sudarshan, "Fine grained authorization through predicated grants," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pp. 1174–1183, IEEE, 2007.

[12] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 224–274, 2001.

[13] R. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pp. 217–228, IEEE, 2005.

[14] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pp. 25–25, IEEE, 2006.

[15] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu, "Utility-based anonymization using local recoding," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 785–790, ACM, 2006.

[16] T. Iwuchukwu and J. Naughton, "K-anonymization as spatial indexing: Toward scalable and incremental anonymization," in *Proceedings of the 33rd international conference on Very large data bases*, pp. 746–757, VLDB Endowment, 2007.

[17] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 49–60, ACM, 2005.

[18] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[19] A. Meyerson and R. Williams, "On the complexity of optimal k-anonymity," in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 223–228, ACM, 2004.

[20] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation algorithms for k-anonymity," *Journal of Privacy Technology*, vol. 2005112001, 2005.

[21] W. Du, D. Eppstein, M. Goodrich, and G. Lueker, "On the approximability of geometric and geographic generalization and the min-max bin covering problem," *Algorithms and Data Structures*, pp. 242–253, 2009.

[22] E. Otoo, D. Rotem, and S. Seshadri, "Optimal chunking of large multidimensional arrays for data warehousing," in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pp. 25–32, ACM, 2007.

[23] A. Frank and A. Asuncion, "UCI Machine Learning Repository," 2010.

[24] B. Steven, A. Trent, G. Katie, G. Ronald, B. S. Matthew, and M. S., "Integrated Public Use Microdata Series: Version 5.0 [machine-readable database]," 2010.

[25] "General Algebraic Modeling System (GAMS)." http://www.gams.com/.

[26] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending query rewriting techniques for fine-grained access control," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 551–562, ACM, 2004.

[27] K. Browder and M. Davidson, "The virtual private database in oracle9ir2," *Oracle Technical White Paper, Oracle Corporation*, vol. 500, 2002.

[28] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi, "Extending relational database systems to automatically enforce privacy policies," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pp. 1013–1022, IEEE, 2005.

[29] S. Chaudhuri, R. Kaushik, and R. Ramamurthy, "Database access control & privacy: Is there a common ground?," in *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR)*, pp. 96–103, 2011.

[30] C. Dwork, "Differential privacy," *Automata, languages and programming*, pp. 1–12, 2006.

[31] N. Li, W. Qardaji, and D. Su, "Provably private data anonymization: Or, k-anonymity meets differential privacy," *Arxiv preprint arXiv:1101.2604*, 2011.

[32] V. Ciriani, S. De Capitani Di Vimercati, S. Foresti, and P. Samarati, "Theory of privacy and anonymity," in *Algorithms and theory of computation handbook*, pp. 18–18, Chapman & Hall/CRC, 2010.

[33] H. Samet, *The design and analysis of spatial data structures*, vol. 85. Addison-Wesley Reading MA, 1990.

[34] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast data anonymization with low information loss," in *Proceedings of the 33rd international conference on Very large data bases*, pp. 758–769, VLDB Endowment, 2007.

[35] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "A framework for efficient data anonymization under privacy and

accuracy constraints," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 2, p. 9, 2009.

[36] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "ireduct: Differential privacy with reduced relative errors," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2011.

APPENDIX A
PROOF OF HARDNESS

Theorem A.1. *Decisional k-anonymity with Imprecision Bounds is NP-complete.*

Proof: The proof is by reduction from Partition:
Partition Given a finite set A and a size function $s(a_i) \in \mathbb{Z}^+$ for each $a_i \in A$. Does there exist a subset, $A^{'} \subseteq A$ such that

$$\underset{a_i\in A'}{} s(a_i) \quad = \quad \underset{a_j\in A-A'}{} s(a_j) \quad ?$$

For each $a_i \in A$, construct multiple tuples that are repeated a number of times equal to count, i.e., forming a multiset, where count is equal to $s(a_i)$. We can equivalently represent these repeated tuples as a set of distinct pairs of the form $(tuple, count)$. This construct is similar to that in [14]. In each pair, the tuple is a point in the d-dimensional unit-hypercube defined by a vector $[0_1, \ldots, 0_{i-1}, 1_i, 0_{i+1}, \ldots, 0_d,]$ (i.e., the i^{th} co-ordinate is 1 and all others are 0). The union of all such pairs is Table T. In this d-dimensional unit-hypercube, construct Query Q so that Q encloses a single $(tuple, count)$ pair.

The partition problem for A can be reduced to the following: *Let $k = \sum \frac{s(a_i)}{2}$, $B_Q = \sum \frac{s(a_i)}{2}$ and qv = 1. Is there a k-anonymous multidimensional partitioning for T such that $imp_Q \leq B_Q$?* We claim that there is a solution to the k-anonymous multidimensional partitioning for T satisfying the imprecision bound for the query Q if and only if there is a solution to the partition problem for A.

Suppose there exists a k-anonymous multidimensional partitioning for T satisfying the imprecision bound for the query Q. The partitions will define two multidimensional regions R_1 and R_2 such that $\sum_{t\in R_1} count(t) = \sum_{t\in R_2} count(t) = k = \sum \frac{s(a_i)}{2}$. The $count(t)$ values in R_1 and R_2 will give the two disjoint subsets of A that define an equal partitioning of A.

In the other direction, suppose there is a solution to the partition problem for A. The solution will define two disjoint subsets A_1 and A_2. From these two subsets, we can find the multidimensional partitions R_1 and R_2 such that $\sum_{t\in R_1} count(t) = \sum_{s(a_i)\in A_1} s(a_i)$ and $\sum_{t\in R_2} count(t) = \sum_{s(a_i)\in A_2} s(a_i)$. The imprecision for the Query Q is less than B_Q as the overlapping partition has size $k = B_Q$.

Finally, a given solution to the decisional k-anonymous multidimensional partitioning problem with imprecision bounds can be verified in polynomial time. All the multidimensional partitions are checked to see if they satisfy the k-anonymity requirement and that the imprecision bound for the query is satisfied. □

Example 6. Let the multiset of size function of set A is $\{1, 2, 2, 3\}$ and the Query Q be on attribute a_1. The $(tuple, count)$ pairs for A are

$$\begin{aligned} t_1 &= ([1,0,0,0],1) \qquad & t_2 &= ([0,1,0,0],2) \\ t_3 &= ([0,0,1,0],2) \qquad & t_4 &= ([0,0,0,1],3) \end{aligned}$$

Then $k = B_Q = \sum \frac{s(a_i)}{2} = 4$ and the solution for partition of A is $A_1 = \{1, 3\}$ and $A_2 = \{2, 2\}$. The corresponding multidimensional partitions for table are 4-anonymous and imprecision for Query Q is 3 which is less than 4.