

Accuracy-Constrained Privacy-Preserving Access Control Mechanism for Relational Data

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

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 user can still compromise the privacy of a person leading to identity disclosure. A PPM can use suppression and generalization of relational data to anonymize and satisfy privacy requirements, e.g., k -anonymity and l -diversity, against identity and attribute disclosure. However, privacy is achieved at the cost of precision of authorized information. In this paper, we propose an accuracy-constrained 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 their 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 users 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 users [2]. This problem has been studied extensively in the area of micro data publishing [3] and privacy definitions, e.g., k -anonymity [2], l -diversity [4], and variance diversity [5]. Anonymization algorithms use suppression and generalization of records to satisfy privacy requirements with minimal distortion of micro data. The

anonymity techniques can be used with an access control mechanism to ensure both security and privacy of the sensitive information. The privacy is achieved at the cost of accuracy and imprecision is introduced in the authorized information under an access control policy.

We use the concept of imprecision bound for each permission to define a threshold on the amount of imprecision that can be tolerated. Existing workload-aware anonymization techniques [5], [6] minimize the imprecision aggregate for all queries and the imprecision added to each permission/query in the anonymized micro data is not known. Making the privacy requirement more stringent (e.g., increasing the value of k or l) results in additional imprecision for queries. However, the problem of satisfying accuracy constraints for individual permissions in a policy/workload has not been studied before. The heuristics proposed in this paper for accuracy-constrained privacy-preserving access control are also relevant in the context of workload-aware anonymization. The anonymization for continuous data publishing has been studied in literature [3]. In this paper the focus is on a static relational table that is anonymized only once. To exemplify our approach, role-based access control is assumed. However, the concept of accuracy constraints for permissions can be applied to any privacy-preserving security policy, e.g., discretionary access control.

Example 1 (Motivating Scenario). Syndromic surveillance systems are used at the state and federal levels to detect and monitor threats to public health [7]. The department of health in a state collects the emergency

- Z. Pervaiz and A. Ghafoor are with the School of Electrical and Computer Engineering and Purdue's Center for Education and Research in Information Assurance and Security (CERIAS). E-mail: zpervaiz@purdue.edu.
- W.G. Aref is with the Department of Computer Science and Purdue's Center for Education and Research in Information Assurance and Security (CERIAS).
- N. Prabhu is with the School of Industrial Engineering, Purdue University, IN 47907.

Manuscript received 1 Oct. 2012; revised 17 Feb. 2013; accepted 11 Apr. 2013; published online xx xx. xxxx.

Recommended for acceptance by E. Ferrari.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-10-0662. Digital Object Identifier no. 10.1109/TKDE.2013.71

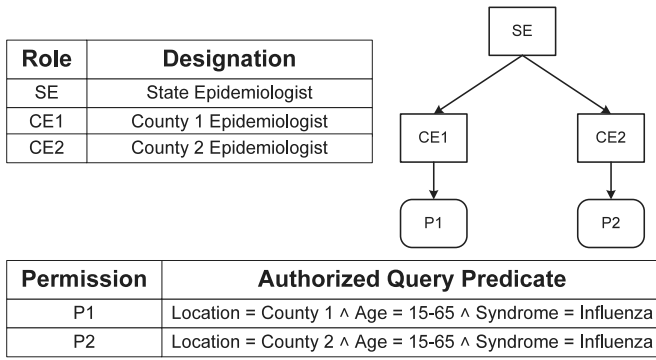


Fig. 1. Access control policy.

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 Fig. 1 that allows the roles to access the tuples under the authorized predicate, e.g., Role CE1 can access tuples under 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 1,000 and quarantine if that number is greater than 3,000 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.

The contributions of the paper are as follows. First, we formulate the accuracy and privacy constraints as the problem of k -anonymous Partitioning with Imprecision Bounds (k -PIB) and give hardness results. Second, we introduce the concept of accuracy-constrained privacy-preserving access control for relational data. Third, we propose heuristics to approximate the solution of the k -PIB problem and conduct empirical evaluation.

The remainder of this paper proceeds as follows. In Section 2, relevant background is discussed. The problem formulation and access control framework are presented in Section 3. Section 4 covers the proposed top-down heuristics for multi-dimensional 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 and privacy definitions based on anonymity are over-viewed. Query

evaluation semantics, imprecision, and the Selection Mondrian algorithm [5] are briefly explained.

Given a relation $T = \{A_1, A_2, \dots, A_n\}$, where A_i is an attribute, T^* is the anonymized version of the relation T . We assume that T is a static relational table. 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 Access Control for Relational Data

Fine-grained access control for relational data allows to define tuple-level permissions, e.g., Oracle VPD [8] and SQL [9]. For evaluating user queries, most approaches assume a Truman model [10]. In this model, a user query is modified by the access control mechanism and only the authorized tuples are returned. Column level access control allows queries to execute on the authorized column of the relational data only [8], [11]. Cell level access control for relational data is implemented by replacing the unauthorized cell values by *NULL* values [12].

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 RBAC model, we assume that the selection predicates on the QI attributes define a permission [11]. UA is a user-to-role ($U \times R$) assignment relation and PA is a role-to-permission ($R \times P$) assignment relation. A role hierarchy (RH) defines an inheritance relationship among roles and is a partial order on roles ($R \times R$) [13]. Each permission defines a hyper-rectangle in the tuple space and all the tuples enclosed by this hyper-rectangle are authorized to the role assigned to the permission. In practice, when a user assigned to a role executes a query, the tuples satisfying the conjunction of the query predicate and the permission are returned [1], [10].

2.2 Anonymity Definitions

In this section, privacy definitions related to anonymity are introduced.

Definition 1 (Equivalence Class (EC)). An equivalence class is a set of tuples having the same QI attribute values.

Definition 2 (k -anonymity Property). A table T^* satisfies the k -anonymity property if each equivalence class has k or more tuples [2].

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 has been proposed [4] and requires that each equivalence

	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 Table

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

class of T^* contain at least l distinct values of the sensitive attribute. For sensitive numeric attributes, an l -diverse equivalence class can still leak information if the numeric values are close to each other. For such cases, variance diversity [5] has been proposed that requires the variance of each equivalence class to be greater than a given variance diversity parameter.

The table 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 table in Fig. 2b is a 2-anonymous and 2-diverse version of table in Fig. 2a. 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

In this section the query predicate evaluation semantics have been discussed. 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:

1. *Uniform*. Assuming the uniform distribution of tuples in the overlapping partitions, include tuples from all partitions according to the ratio of overlap between the query and the partition. Query evaluation under this option 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 semantics to compare anonymity techniques over selection tasks [6], [14]. However, the choice of the sensitive

attribute value for the selected tuples from an overlapping partition is not defined under uniform semantics. For access control, a tuple's QI attribute values along with the sensitive attribute value need to be returned.

2. *Overlap*. Include all tuples in all partitions that overlap the query region. This option will add false positives to the original query result.
3. *Enclosed*. Discard all tuples in all partitions that partially overlap the query region. This option will have false negatives with respect to the original query result.

The imprecision under any query evaluation scheme is reduced if the number of tuples in the partitions that overlap the query region can be minimized. For the remainder of this paper, we assume Overlap semantics. The imprecision quality metric definition using Overlap semantics is as follows [5]:

Definition 3 (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} ,*

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

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

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

Example 2. Consider a range Query $Q_1(0-25, 5-20)$ for the table given in Fig. 2. $|Q_1(T)| = 2$ as tuples 1 and 4 in Fig. 2a satisfy the query. $|Q_1(T^*)| = 5$ as the first two equivalence classes given in Fig. 2b overlap the query range. Then, the query imprecision for Q_1 is 3 according to Equation (1).

2.4 Top Down Selection Mondrian

Top Down Selection Mondrian (TDSM) algorithm is proposed by LeFevre et al. [5], [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 [15]. 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 [5], 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 [5] and is $\mathcal{O}(d|Q|n \lg n)$, 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 $\lg n$ and the work done at each level is $d|Q|n$. The partitions created by TDSM have dimensions along the median of the parent partition. A compaction procedure has been proposed in [6] 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. In Section 5, compaction is carried out for all the algorithms and then the results are compared.

3 ANONYMIZATION WITH IMPRECISION BOUNDS

In this section, we formulate the problem of k -anonymous Partitioning with Imprecision Bounds and present an accuracy-constrained privacy-preserving access control framework.

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 $\{v_1^i, \dots, v_d^i\}$, 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 $t_i \in D_{QI_1} \times \dots \times D_{QI_d}$. Any d -dimensional Partition P_i of the QI attribute domain space can be defined as a d -dimensional vector of closed intervals $\{I_1^{P_i}, \dots, I_d^{P_i}\}$. 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 $\{v_1, \dots, v_d\}$ for each tuple is replaced by the intervals $\{I_1^{P_i}, \dots, I_d^{P_i}\}$ 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 $\{QI_1, \dots, QI_d\}$. A query defines a space in the domain of quasi-identifier attributes $D_{QI_1} \times \dots \times D_{QI_d}$ 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 $\{I_1^{Q_i}, \dots, I_d^{Q_i}\}$. 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 4 (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 administrator.

Example 3. Assume two range queries as given in Fig. 3. The queries are the shaded rectangles with solid lines while the partitions are the regions enclosed by

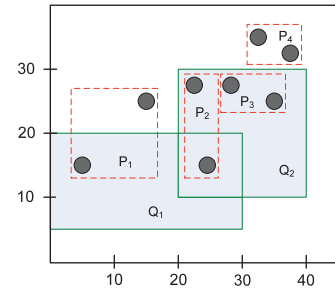


Fig. 3. Anonymization satisfying imprecision bounds.

rectangles with dashed lines. The imprecision bounds for Queries Q_1 and Q_2 are preset to 2 and 0. The partitioning given in Fig. 2b does not satisfy the imprecision bounds. However, the partitioning given in Fig. 3 satisfies the bounds for Queries Q_1 and Q_2 as the imprecision for Q_1 and Q_2 is 2 and 0, respectively.

Definition 5 (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.

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

Definition 6 (Partition Imprecision Cost (PIC)). The partition imprecision cost is a vector $\{ic_{P_1}^{Q_1}, \dots, ic_{P_n}^{Q_n}\}$, 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.,

$$ic_{P_i}^{Q_j} = |P_i - Q_j|, \quad (3)$$

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

$$imp_{Q_j} = \sum_{P_i \in P} ic_{P_i}^{Q_j}.$$

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 7 (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.

Example 4. A comparison of median cut and query cut is given in Fig. 4 for 3-anonymity. The rectangle with solid lines represents Query Q_1 . While, the rectangles with dotted lines represent partitions. In Fig. 4a 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 Fig. 4b the imprecision is reduced to zero as partitions are either non-overlapping or fully enclosed inside the query region.

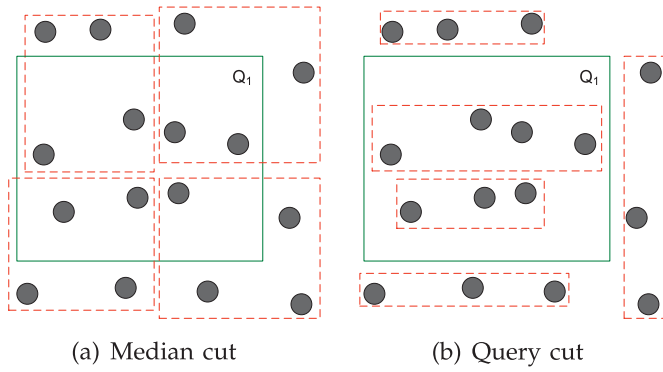


Fig. 4. Comparison of median and query cut.

3.2 The k -PIB Problem

The optimal k -anonymity problem has been shown to be NP-complete for suppression [16] and generalization [17]. 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 violates 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 8 (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 multi-dimensional partitioning for T such that the size of every multidimensional partition R_i is greater than or equal to k and the number of queries violating imprecision bounds is less than the positive constant qv ?

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

Proof. Refer to Appendix, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.71>. \square

3.3 Accuracy-Constrained Privacy-Preserving Access Control

An accuracy-constrained privacy-preserving access control mechanism, illustrated in Fig. 5 (arrows represent the direction of information flow), is proposed. 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 permissions in the access control policy are based on selection predicates on the QI attributes. The policy administrator defines the permissions along with the imprecision bound for each permission/query, user-to-role assignments, and role-to-permission assignments [18]. The specification of the imprecision bound ensures that the authorized data has the desired level of accuracy. The imprecision bound information is not shared with the users because knowing

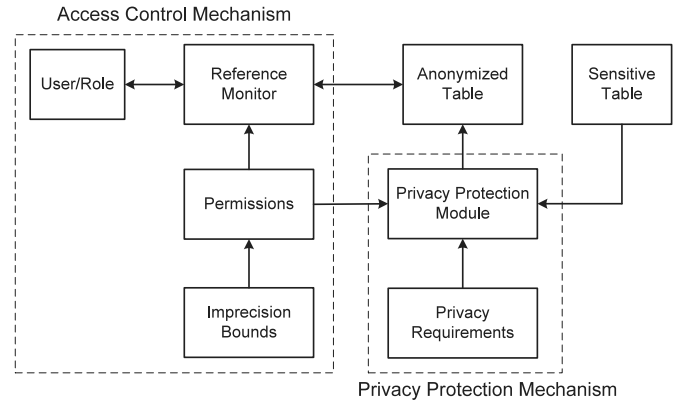


Fig. 5. Accuracy-constrained privacy-preserving access control mechanism.

the imprecision bound can result in violating the privacy requirement. The privacy protection mechanism is required to meet the privacy requirement along with the imprecision bound for each permission.

3.3.1 Access Control Enforcement

The exact tuple values in a relation are replaced by the generalized values after the anonymization. In this case, access control enforcement over the generalized data needs to be defined. In this section, we discuss the Relaxed and Strict access control enforcement mechanisms over anonymized data. The access control enforcement by reference monitor can be of the following two types:

1. *Relaxed.* Use overlap semantics to allow access to all partitions that are overlapping the permission.
2. *Strict.* Use enclosed semantics to allow access to only those partitions that are fully enclosed by the permission.

Both schemes have their own pros and cons. Relaxed enforcement violates the authorization predicate by giving access to extra tuples but is beneficial for applications where low cost of a false alarm is tolerable as compared to the risk associated with a missed event. Examples include epidemic surveillance and airport security. On the other hand, strict enforcement is suitable for applications where a high risk is associated with a false alarm as compared to the cost of a missed event. An example is a false arrest in case of shoplifting. In this paper, the focus is on relaxed enforcement. However the proposed methods for anonymization are also valid for strict enforcement because the proposed heuristics reduce the overlap between partitions and queries. We further assume that under relaxed enforcement if the imprecision bound is violated for a permission then that permission is not assigned to any role.

3.3.2 Probabilistic Analysis for Access Control Enforcement

In this section, the relaxed enforcement of access control is analyzed probabilistically. The access control policy administrator sets the imprecision bound B_{Q_i} for each query, and requires that the imprecision bound for the

least number of queries be violated by PPM. The policy administrator might revise the imprecision bounds for queries and further relax the access control policy if it is known with a high probability that a large number of queries will violate the bounds and access requests for roles will be denied. From this perspective, we are interested in answering the following two questions:

1. What is the average imprecision for a given query?
2. Given a set of queries with imprecision bounds, how many queries are expected to violate the bounds?

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. [19], where they find overlapping intervals in each dimension and then compute 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 the domain space and then divide this length of the 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. Let I_{Q_j} be a non-negative random variable that denotes the query imprecision. Then, the expected imprecision for a query Q_j is

$$E(I_{Q_j}) \leq \left[\left(\prod_{i=1}^d \left\lceil \frac{l_i^{Q_j} + l_i^{P_e}}{l_i^{P_e}} \right\rceil \right) * |P_e| \right] - |Q_j|. \quad (4)$$

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.

Given an imprecision bound B_{Q_i} for a Query Q_i , for the second question, we are interested in finding the expected number of queries that will violate the bounds. Let X_1, \dots, X_n be a set of independent random variables such that $Pr(X_i = 1) = p_i$ and $Pr(X_i = 0) = 1 - p_i$ where, $0 \leq p_i \leq 1$. X_i is a random variable that is equal to 1 if the Query Q_i violates the imprecision bound B_{Q_i} otherwise is equal to 0. The total number of queries violating their imprecision bounds is $X = \sum_{i=1}^n X_i$. X_1, \dots, X_n are called a *Poisson trial* and follow a *Poisson binomial* distribution. The expected number of queries violating their imprecision bounds $E[X] = \mu = \sum_{i=1}^n p_i$ [20]. Dependencies exist among the queries but for our analysis we assume that queries are independent.

Theorem 3.3. Let I_{Q_i} be a non-negative random variable that denotes the query imprecision. Let X_1, \dots, X_n be an independent Poisson trial, where X_i is a random variable that is equal to 1 if a query, say Q_i , violates the imprecision bound B_{Q_i} otherwise is equal to 0. For $X = \sum_{i=1}^n X_i$ and $B_{Q_i} > 0$, we have

$$E[X] = \sum_{i=1}^n p_i \leq \sum_{i=1}^n \frac{E(I_{Q_i})}{(B_{Q_i} + 1)}. \quad (5)$$

Proof. Refer to Appendix, available in the online supplemental material. \square

4 HEURISTICS FOR PARTITIONING

In this section, three algorithms based on greedy heuristics are proposed. All three algorithms are based on kd-tree construction [15]. 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 [6].

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 Fig. 4. 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 $ic_{CP_i}^{QO_j} > 0$
- 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 **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 $\mathcal{O}(d|Q|^2n^2)$. \square

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 $ic_{P_i}^{Q_j}$ 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.

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.

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

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 data sets (greater than 10 million tuples). In the Top-Down Heuristic 3 algorithm (TDH3, for short), we modify TDH2 so that the time complexity of $\mathcal{O}(d|Q|n \lg n)$ 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 $ic_{CP_i}^{Q_j} > 0$
 - 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 $\mathcal{O}(d|Q|nlgn)$. \square

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 data sets for the empirical evaluation of the proposed heuristics. The first data set is the Adult data set from the UC Irvine Machine Learning Repository [21] having 45,222 tuples and is the de facto benchmark for k -anonymity research. The attributes in the Adult data set are: Age, Work class, Education, Marital status, Occupation, Race, and Gender. The second data set is the Census data set [22] from IPUMS. This data set is extracted for Year 2001 using attributes: Age, Gender, Marital status, Race, Birth place, Language, Occupation, and Income. The size of the data set 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 data set) 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 (9,999,999 in the data set) are removed, which leaves about 950k tuples.

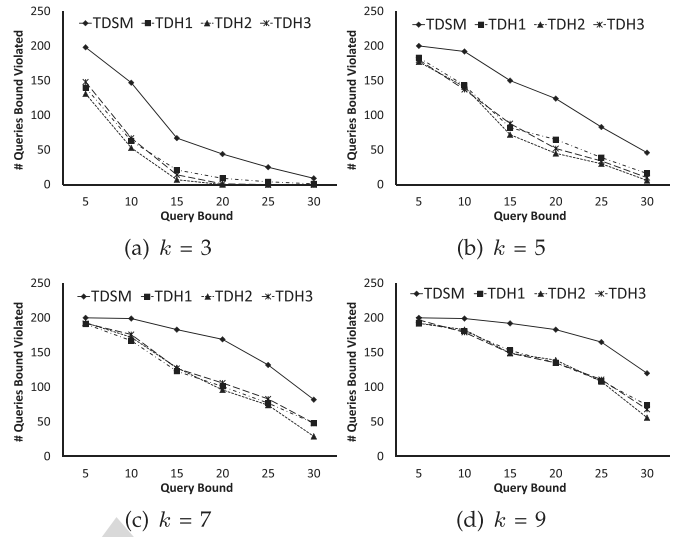


Fig. 6. No of queries violating bounds for k -anonymity for the Adult data set.

We use 200 and 500 queries generated randomly as the workload/permissions for the Adult data set and Census data set, 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 and Naughton [6]. 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 data sets is set to range from 0.5 to 5 percent. For the first workload, if the query output is between 500 to 5,500 tuples for the Adult data set and 1,000 to 50,000 for the Census data set, the query is added to the workload. For the second workload (we will refer to this workload as the uniform query workload) this range (1,000 to 50,000 for Census data set) is divided into ten equal intervals and we add only 50 queries from each interval to the workload. Similarly, for the Adult data set, 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 data set is 3,000 and for the Census data set 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 administrator. 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. Further, as no real relational policy data is available, we believe this approach can allow researchers to reproduce our workload and compare their results with the approaches presented in this paper.

For the k -anonymity experiments, we fix the value of k and change the query imprecision bounds from 5 to 30 percent 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 Fig. 6 for the Adult data set for k values of 3, 5, 7 and 9. Heuristic TDH2 has the least number of query bound violations and is better than

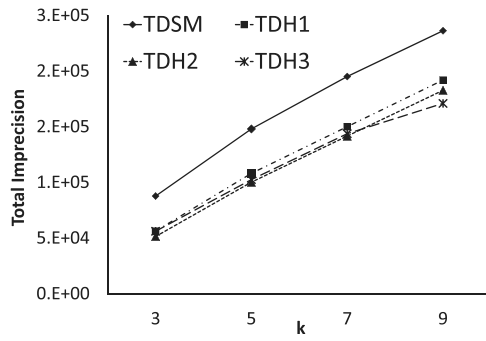


Fig. 7. Total imprecision for all queries for the Adult data set.

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 violating 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 Fig. 7, 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 data set.

For k -anonymity, the number of queries for which the imprecision bound is violated is given in Fig. 8 for the Census data set using the uniform query workload of 500 queries. The results have the same behavior as that for the Adult data set. In both cases, TDH2 has the lowest number of queries violating the imprecision bounds. The sum of imprecision for all queries is given in Fig. 9, where TDH2 also has the lowest total imprecision for all values of k . In Fig. 8, the total number of violated queries is given. So, in Fig. 10, we plot the number of queries against the margin by which they violate the query bound (Imprecision bound is set as 25 percent 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 percent and greater than 100 percent of the bound. In Section 6, an algorithm is proposed to realign the

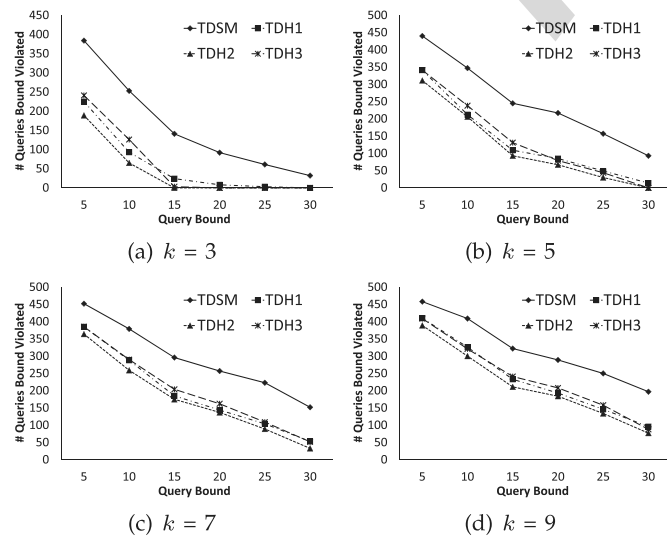
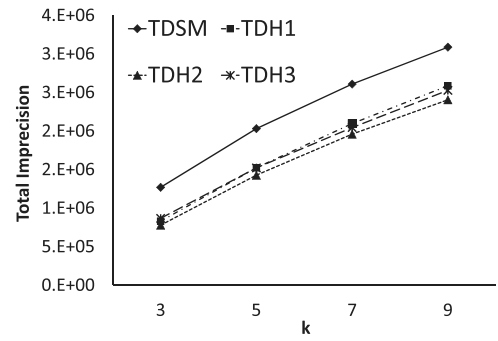
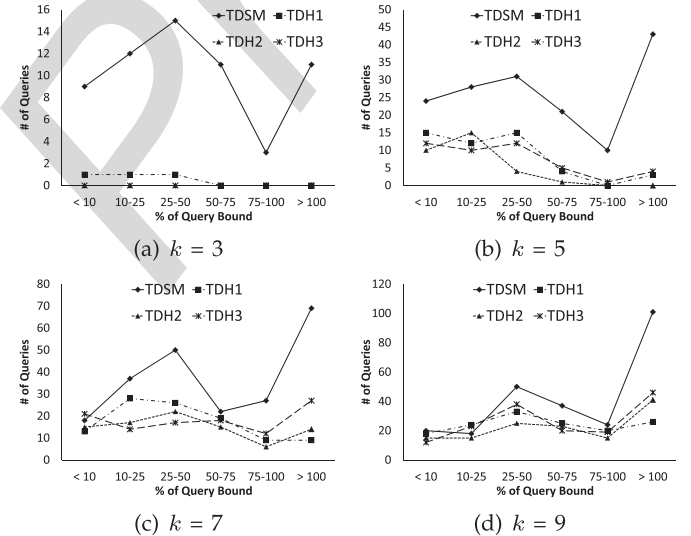
Fig. 8. No. of queries violating bounds for k -anonymity for the Census data set.

Fig. 9. Total imprecision for all queries for the Census data set.

output partitions to satisfy the imprecision bounds of queries that violate the bound by a less than 10 percent margin. The reason for using the uniform query workload (50 randomly selected queries from each size range having cardinality between 0.5 to 5 percent of the data set) 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 Fig. 11, 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. 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 percent 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 Fig. 12. The

Fig. 10. Distribution of queries (wrt bound) violating bound at 25 percent for k -anonymity for the Census data set.

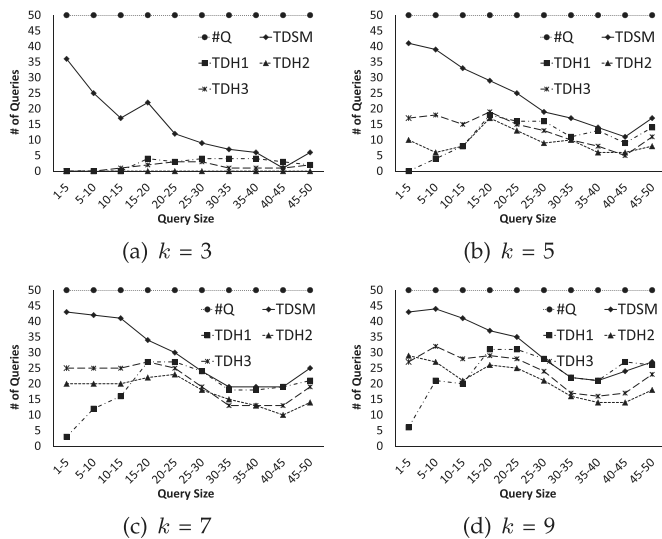


Fig. 11. Distribution of queries (wrt size) violating bound at 15 percent for k -anonymity for the Census data set.

results show that TDH2 violates the bound for a less number of queries for l -diversity.

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 data set. For a variance diversity value, we change the query imprecision bounds from 5 to 30 percent and find the number of queries whose bounds are violated by each algorithm. The results for variance diversity are given in Fig. 13. For variance diversity, TDH2 gives the best results.

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 1,024 for a k value of 5 and a query imprecision bound of 30 percent. Observe in Fig. 14 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 1,000 new random queries and compare the number of queries satisfied at 30 percent imprecision bound by each algorithm. The results are given in Fig. 15. Observe that the performance of all the algorithms is similar. The slightly better results

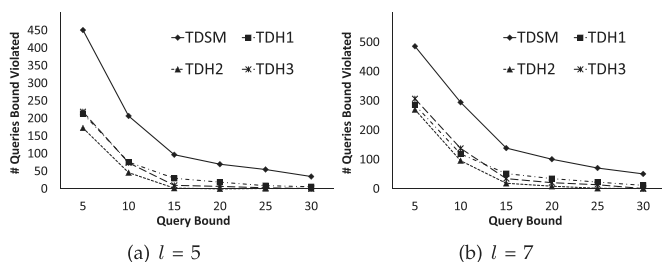


Fig. 12. Number of queries violating bound for l -diversity for the Census data set.

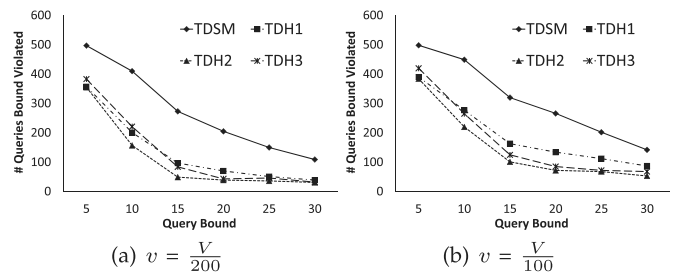


Fig. 13. Number of queries violating bound for variance-diversity for the Census data set.

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.

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 data set. We use three attributes (Work Class, Marital Status, and, Race) and pick 1,000 tuples randomly from the Adult data set. The heuristic algorithms are executed using a workload of 1,000 randomly selected queries with an imprecision bound of 20 percent 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 General Algebraic Modeling System (GAMS) 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 Fig. 16. 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 Fig. 17. Here, 1,000 tuples with two attributes are randomly selected (Normal distribution with $\mu = 50$, $\sigma = 10$, and cardinality = 100). 10 random queries are also selected (Query selectivity is from 10 to 50 percent) and

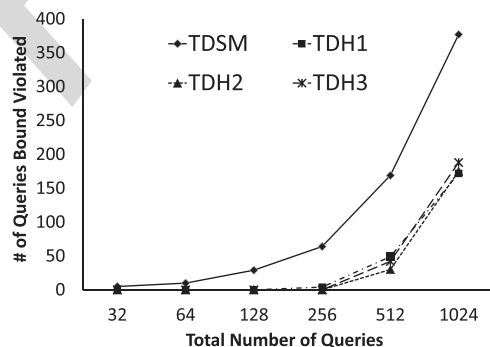


Fig. 14. Varying the size of given query workload for the Census data set.

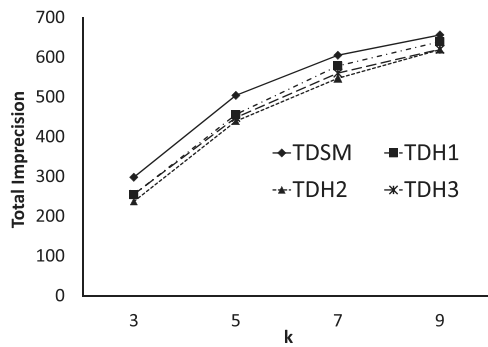


Fig. 15. Performance for a different query workload for the Census data set.

the query imprecision bound is set to 10 percent 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 $k=5$. Observe that in Fig. 17, 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.

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 Fig. 10. In this repartitioning step, we consider only the first two groups of queries that fall within 10 percent and 10-25 percent 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 SQ . 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 percent 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 SQ . 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 SQ . The proposed

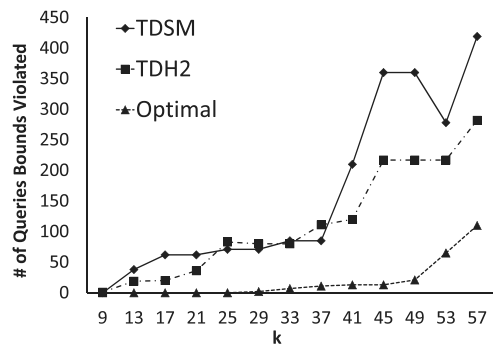


Fig. 16. Comparison with optimal solution.

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.

Algorithm 4: Repartitioning

Input : T, k, Q, P , and B_q

Output: P

- 1 Initialize SQ, CQ , and CP
 - 2 Add $q \in Q$ satisfying bound to SQ
 - 3 Add $q \in Q$ violating bound by 10% to Candidate Query set(CQ)
 - 4 Add all sibling leaf node pairs having $\sum_{q \in CQ} (ic_{P_i}^{q_i} + ic_{P_{i+1}}^{q_i}) > 0$ to Candidate Partition(CP)
 - 5 **for** ($CP_i \in CP$) **do**
 - 6 Merge the first pair CP_i and CP_{i+1}
 - 7 Select q from CQ with the least imprecision greater than the imprecision bound
 - 8 Create the candidate cuts in each dimension
 - 9 Select the cut and the dimension satisfying all $q \in SQ$ with the minimum imprecision $\forall q \in CQ$
 - 10 **if** (*feasible cut found*) **then**
 - 11 Remove CP_i and CP_{i+1} from CP and P
 - 12 Add new partitions to P
 - 13 **for** ($q \in CQ$) **do**
 - 14 **if** ($Imp_q < B_q$) **then**
 - 15 Remove q from CQ and add to SQ
 - 16 **return** (P)
-

In the experiments, we set the value of k to 5 and 7 with a query imprecision bound of 30 percent of the query size. The results for repartitioning are given in Fig. 18. TDH2p and TDH3p are the results after the repartitioning step. Observe that most of the queries in the 10 percent group have been satisfied, while for the 10-25 percent group, some of these have been satisfied while the others have moved into the first group. 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.

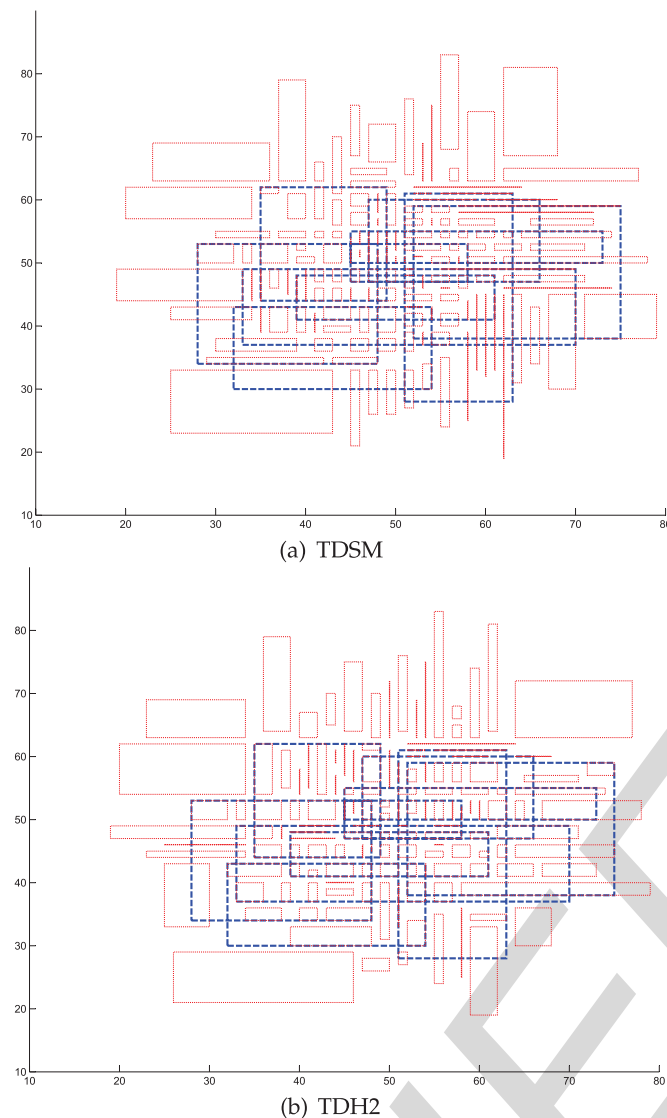


Fig. 17. Anonymization for two attributes with discrete normal distribution ($\mu = 50, \sigma = 10$).

7 RELATED WORK

Access control mechanisms for databases allow queries only on the authorized part of the database [8], [10]. 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 [23]. 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 [24]. They use the definition of *differential privacy* [25] 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. [26] that after sampling, k -anonymity offers similar privacy guarantees as those of *differential privacy*. The proposed accuracy-constrained privacy-preserving access control framework allows the access control administrator to specify imprecision constraints that the

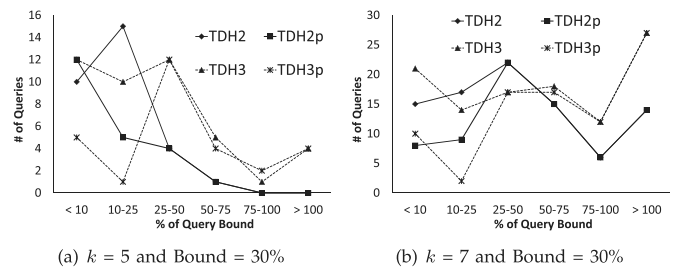


Fig. 18. Improvements after repartitioning for k -anonymity for the Census data set.

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 a recent survey paper [3]. Workload-aware anonymization is first studied by LeFevre et al. [5]. 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 and Naughton have proposed an R^+ -tree based anonymization algorithm [6]. 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 [27]. They also introduce the problem of accuracy-constrained anonymization for a given bound of acceptable information loss for each equivalence class [28]. Similarly, Xiao et al. [29] 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. [5] and introduce the constraint of imprecision bound for each query in a given query workload.

8 CONCLUSIONS

An accuracy-constrained privacy-preserving access control framework for relational data 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 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 cell level access control.

ACKNOWLEDGMENTS

This research was partially supported by the US National Science Foundation (NSF) Grants IIS-1117766, IIS-0964639, and IIS-0811954.

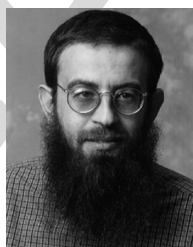
REFERENCES

- [1] E. Bertino and R. Sandhu, "Database Security-Concepts, Approaches, and Challenges," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 1, pp. 2-19, Jan.-Mar. 2005.
- [2] P. Samarati, "Protecting Respondents' Identities in Microdata Release," *IEEE Trans. Knowledge and Data Eng.*, vol. 13, no. 6, pp. 1010-1027, Nov. 2001.
- [3] B. Fung, K. Wang, R. Chen, and P. Yu, "Privacy-Preserving Data Publishing: A Survey of Recent Developments," *ACM Computing Surveys*, vol. 42, no. 4, article 14, 2010.
- [4] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkatasubramanian, "L-Diversity: Privacy Beyond k-anonymity," *ACM Trans. Knowledge Discovery from Data*, vol. 1, no. 1, article 3, 2007.
- [5] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Workload-Aware Anonymization Techniques for Large-Scale Datasets," *ACM Trans. Database Systems*, vol. 33, no. 3, pp. 1-47, 2008.
- [6] T. Iwuchukwu and J. Naughton, "K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization," *Proc. 33rd Int'l Conf. Very Large Data Bases*, pp. 746-757, 2007.
- [7] J. Buehler, A. Sonrick, 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.
- [8] K. Browder and M. Davidson, "The Virtual Private Database in oracle9i2," Oracle Technical White Paper, vol. 500, 2002.
- [9] A. Rask, D. Rubin, and B. Neumann, "Implementing Row-and Cell-Level Security in Classified Databases Using SQL Server 2005," MS SQL Server Technical Center, 2005.
- [10] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending Query Rewriting Techniques for Fine-Grained Access Control," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 551-562, 2004.
- [11] S. Chaudhuri, T. Dutta, and S. Sudarshan, "Fine Grained Authorization through Predicated Grants," *Proc. IEEE 23rd Int'l Conf. Data Eng.*, pp. 1174-1183, 2007.
- [12] K. LeFevre, R. Agrawal, V. Ercegovic, R. Ramakrishnan, Y. Xu, and D. DeWitt, "Limiting Disclosure in Hippocratic Databases," *Proc. 30th Int'l Conf. Very Large Data Bases*, pp. 108-119, 2004.
- [13] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," *ACM Trans. Information and System Security*, vol. 4, no. 3, pp. 224-274, 2001.
- [14] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Mondrian Multidimensional K-Anonymity," *Proc. 22nd Int'l Conf. Data Eng.*, pp. 25-25, 2006.
- [15] J. Friedman, J. Bentley, and R. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Trans. Mathematical Software*, vol. 3, no. 3, pp. 209-226, 1977.
- [16] A. Meyerson and R. Williams, "On The Complexity of Optimal k-Anonymity," *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems*, pp. 223-228, 2004.
- [17] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation Algorithms for k-Anonymity," *J. Privacy Technology*, vol. 2005112001, 2005.
- [18] R. Sandhu and Q. Munawer, "The Arbac99 Model for Administration of Roles," *Proc. 15th Ann. Computer Security Applications Conf.*, pp. 229-238, 1999.
- [19] E. Otoo, D. Rotem, and S. Seshadri, "Optimal Chunking of Large Multidimensional Arrays for Data Warehousing," *Proc. ACM 10th Int'l Workshop on Data Warehousing and OLAP*, pp. 25-32, 2007.
- [20] W. Hoeffding, "On the Distribution of the Number of Successes in Independent Trials," *The Annals of Math. Statistics*, vol. 27, no. 3, pp. 713-721, 1956.

- [21] A. Frank and A. Asuncion, "UCI Machine Learning Repository," 2010.
- [22] B. Steven, A. Trent, G. Katie, G. Ronald, B.S. Matthew, and M. Sobek, "Integrated Public Use Microdata Series: Version 5.0 [Machine-Readable Database]," 2010.
- [23] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi, "Extending Relational Database Systems to Automatically Enforce Privacy Policies," *Proc. IEEE 21st Int'l Conf. Data Eng.*, pp. 1013-1022, 2005.
- [24] S. Chaudhuri, R. Kaushik, and R. Ramamurthy, "Database Access Control & Privacy: Is There a Common Ground?" *Proc. Fifth Biennial Conf. Innovative Data Systems Research (CIDR)*, pp. 96-103, 2011.
- [25] C. Dwork, "Differential Privacy," *Proc. 33rd Int'l Colloquium Automata, Languages and Programming*, pp. 1-12, 2006.
- [26] N. Li, W. Qardaji, and D. Su, "Provably Private Data Anonymization: Or, k-Anonymity Meets Differential Privacy," *Arxiv preprint arXiv:1101.2604*, 2011.
- [27] S. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast Data Anonymization with Low Information Loss," *Proc. 33rd Int'l Conf. Very Large Data Bases*, pp. 758-769, 2007.
- [28] S. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "A Framework for Efficient Data Anonymization Under Privacy and Accuracy Constraints," *ACM Trans. Database Systems*, vol. 34, no. 2, article 9, 2009.
- [29] X. Xiao, G. Bender, M. Hay, and J. Gehrke, "Ireduct: Differential Privacy with Reduced Relative Errors," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 2011.



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



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



Arif Ghafoor is currently a professor in the School of Electrical and Computer Engineering at Purdue University. He has been actively engaged in research on multimedia information systems, database security, and parallel and distributed computing. He is a fellow of the IEEE.



Nagabhushana Prabhu received the PhD degree in computer science from NYU and the PhD degree in theoretical physics from the MIT. He is a professor in the School of Industrial Engineering at Purdue University. His research interests include optimization, high-energy physics, and computational oncology.

Queries to the Author

- Q1. Please check whether the affiliations of all the authors are ok as set?
- Q2. Please provide the page range in Reference [17].
- Q3. Please supply URL for the repository in Reference [22].
- Q4. Please provide the name of the journal the article was published in.

IEEE
Proof