A Privacy-Preserving Index for Range Queries

Bijit Hore, Sharad Mehrotra, Gene Tsudik

University of California, Irvine
Irvine, CA 92697
{bhore,sharad,gts}@ics.uci.edu

Abstract

Database outsourcing is an emerging data management paradigm which has the potential to transform the IT operations of corporations. In this paper we address privacy threats in database outsourcing scenarios where trust in the service provider is limited. Specifically, we analyze the data partitioning (bucketization) technique and algorithmically develop this technique to build privacy-preserving indices on sensitive attributes of a relational table. Such indices enable an untrusted server to evaluate obfuscated range queries with minimal information leakage. We analyze the worst-case scenario of *inference attacks* that can potentially lead to breach of privacy (e.g., estimating the value of a data element within a small error margin) and identify statistical measures of data privacy in the context of these attacks. We also investigate precise privacy guarantees of data partitioning which form the basic building blocks of our index. We then develop a model for the fundamental *privacy-utility* tradeoff and design a novel algorithm for achieving the desired balance between privacy and utility (accuracy of range query evaluation) of the index.

1 Introduction

The recent explosive increase in the Internet usage, coupled with advances in software and networking, has resulted in organizations being able to easily share data for a variety of purposes. This has given rise to a set of new and interesting computing paradigms. In the database context, one such paradigm is *Database-as-a-Service* (DAS)[4, 6, 7] in which organizations outsource data management to a service provider.

In the DAS model, since data is stored at the service provider (that may not be fully trusted) many new security and privacy challenges arise. Most approaches to DAS define a notion of a *security perimeter* around the data owner. The environment within the perimeter is trusted whereas the environment outside the perimeter is not. For instance, in [4], the client (which is also the data owner) is trusted, while the server (service provider) is not. In [6], a smart card solution is considered in which owners access data using a client terminal supporting smart card devices. The client's smart-card devices and terminals are within the security perimeter, whereas the large data store is considered to be outside. [7] considers a DAS architecture in which a secure co-processor resides alongside the server. The secure co-processor is within the perimeter, while the server which supports a large scale storage is considered to be outside the perimeter.

Query processing in the DAS model: In each of the DAS models considered above, data is stored in an encrypted form outside the security perimeter but accessed from within. Usually, data is accessed through a client query Q. A direct way to support Q is to transfer data from the untrusted servers to the trusted environment within the security perimeter. Once within the security perimeter, data can be decrypted and the query predicates evaluated. Such an approach, however, mitigates many of the primary advantages of DAS where we are interested in outsourcing not just the storage but also database operations. The alternative approach is to split the query Q into two components: $Q_{sec} + Q_{insec}$, where Q_{insec} executes at the server on the encrypted representation to compute a (superset of) results for Q and Q_{sec} executes within the security perimeter to filter out the false positives. The objective is to push as much work as possible for evaluating Q to the service provider.

Techniques to split query processing between the client and server in the context of DAS has been explored for various classes of queries. For example, *bucketization* (data partitioning) technique [4] to support

**Proceedings of the 30th VLDB Conference,
Toronto, Canada, 2004**

range queries, *deterministic encryption* [4, 6] for join queries, as well as, several other methods [5, 1] have been proposed. The objective of these are to push as much work as possible for evaluating the queries to the service provider.

While techniques to support various types of queries in the DAS model have been developed, much of this work is *ad-hoc* in nature. For instance, it lacks an in-depth analysis of the level of privacy afforded under various attack scenarios. This is challenging, specially given that there is no agreed upon definition or established way to reason about privacy. Privacy, in general, depends upon the user specification, nature of data and the application context. Existing research also lacks analysis of the privacy-utility trade-offs inherent to any privacy-preserving data processing system.

In this paper, we focus on range queries and a bucketization based approach proposed in [4] to support them in the DAS model. In the bucketization approach, an attribute domain is partitioned into a set of buckets each of which is identified by a tag. These bucket tags are maintained as an index (referred to as **crypto-index**) and are utilized by the server to process the queries. Our goal in this paper is to characterize the privacy threats arising from the creation of bucketization-based indices to support range queries. Furthermore, we aim to design algorithms to explore privacy-efficiency trade-offs for the bucketization schemes. Specifically, this paper makes the following contributions:

1. Design of an optimal (that maximizes the accuracy of range queries) solution for data bucketization.

2. Identification of privacy measures most relevant in the DAS model and analysis of privacy levels achieved for any instance of bucketized data[1].

3. Development of a novel privacy-preserving re-bucketization technique that yields bounded overhead (due to commensurately reduced accuracy) while maximizing the defined notions of privacy.

The rest of this paper is organized as follows. Section 2 briefly discusses range queries in [4] and addresses relevant privacy issues. Section 3 develops an optimal data bucketization algorithm which maximizes efficiency of crypto-indices. Section 4 identifies statistical metrics of privacy relevant to the context. Next, section 5 introduces our new re-bucketization technique and section 7 discusses privacy issues for the multi-attribute case. Experimental results are reported in section 8 and related work is over viewed in section 9. The paper concludes in Section 10 and 11 with the summary and future work issues.

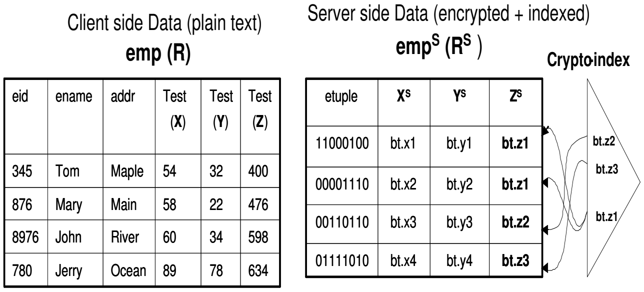
2 Preliminaries

We begin with the brief overview of the DAS model from [4]. This setting involves clients (data owners) and servers (database service providers). Clients to do not trust servers with data contents and encrypt the outsourced data before storing it at the server. Specifically, each data tuple (record or row) of a relational table is stored at the server as an encrypted unit, the so-called *etuple*.

However, since virtually no useful database operations can be performed over encrypted data, the DAS model involves creation of crypto-indices over sensitive attributes[2] which are expected to appear in queries. Multiple crypto-indices may be created over each attribute to support different kinds of SQL queries. The objective is to maximize the amount of query processing done by the server (without, of course, decrypting any etuples) while minimizing work for the client. At query execution time, instead of the actual cleartext attribute values, crypto indices are used for filtering out tuples for query predicates that involve sensitive attributes.

We focus on crypto-indices designed to support range queries. The technique in [4] involves partitioning (*bucketizing*) each attribute domain into a finite number of regions (in an equi-depth or equi-width manner) and assigning each region a unique random *tag* (bucket-id). Subsequently, the cleartext of sensitive attributes of each tuple is essentially suppressed and only identified by its corresponding bucket-id, for each crypto-index built upon that attribute. If the original table was R, this results in a new server-side table R^S, containing etuples and corresponding bucket-ids. An example is given below:

Example 1: Consider the tables in figure 1, where the left side is the original table and the right is its encrypted version. For each sensitive attribute: X, Y and Z, a separate crypto-index is created. The client uses *meta-data* (stored within the secure perimeter) to translate normal database queries into server-side queries. The latter can only use indexing information


Client side Data (plain text)
emp (R)
Server side Data (encrypted + indexed)
emp^S (R^S)
Crypto-index
bt.z2
bt.z3
bt.z1

eid	ename	addr	Test (X)	Test (Y)	Test (Z)
345	Tom	Maple	54	32	400
876	Mary	Main	58	22	476
8976	John	River	60	34	598
780	Jerry	Ocean	89	78	634

etuple	X^S	Y^S	Z^S
11000100	bt.x1	bt.y1	**bt.z1**
00001110	bt.x2	bt.y2	**bt.z1**
00110110	bt.x3	bt.y3	**bt.z2**
01111010	bt.x4	bt.y4	**bt.z3**

Figure 1: Representation of table on the server

[1] In fact, we offer a worst-case analysis where the adversary is assumed to know the entire bucketization scheme.

[2] Crypto-indices corresponding to non-sensitive attributes are the same as in any normal index data structure.

in R^S (i.e., the columns X^S, Y^S, Z^S). For example, an SQL query:

> **Select** *ename, addr* **from** R **where** $R.Z \geq 450$
> is translated into:
> **Select** *etuple* **from** R^S **where** $R^S.Z^S$ = bt.z1
> $\vee$ bt.z2 $\vee$ bt.z3

where bt.z1, bt.z2 and bt.z3 refer to **bucket-tags** of the buckets created on attribute Z. $\diamondsuit$

It is evident that this approach often results in the query reply containing a superset of records desired by the client. To filter out superfluous data, the client needs to post-process the reply: decrypt each etuple and apply the original query criteria to the cleartext. It is easy to make the following observation:

Observation 1 *Allocating a large number of buckets to crypto-indices increases query precision but reduces privacy. On the other hand, a small number of buckets increases privacy but adversely affects performance.*

The goal of the client is thus twofold:
1) Server Efficiency: maximize the server-side accuracy of range query evaluation. Higher efficiency results in lower server-client communication overhead and lower post-processing costs for the client.
2) Maximum Privacy: minimize the information revealed to the server through the crypto-indices. In other words, maximize data privacy. (What constitutes "information" in this context is clarified in Section 4) below).

We now turn to the problem of optimal bucketization of an attribute domain.

3 Optimal Buckets for Range Queries

As noted in the previous section, the problem of optimal bucketization of attribute domains was not addressed in [4]. In this section we develop such an algorithm.

For simplicity of analysis, we will restrict our attention to building crypto-indices over numeric attributes from a discrete domain like **Z** (set of non-negative integers). But the algorithm applies to the real domain as well. Also we should point out here, that the method by which crypto-indices are actually implemented, is immaterial to us. In reality, any simple data structure that can swiftly retrieve all tuples belonging to any bucket, will suffice. The notion of efficiency that we are concerned with, is only dictated by how the data is partitioned into these buckets.

3.1 Problem Statement

We start by defining the optimal bucketization problem. (Refer to the table 1 for notations.)

Problem: 3.1 *Given an input relation $R = (V, F)$ (where V is the set of distinct numeric values appearing at least once in the column and F is the set of corresponding frequencies), a query distribution P (defined on the set of all range queries, Q) and the maximum number of buckets M, partition R into at most M buckets such that the total number of* false positives *over all possible range queries (weighted by their respective probabilities) is minimized.*

V_{min}	minimum possible value for a given attribute
V_{max}	maximum possible value for a given attribute
N	number of possible distinct attribute values; $N = V_{max} - V_{min} + 1$
R	relation (in cleartext), $R = (V, F)$
$\|R\|$	number of tuples in R (i.e. size of table)
V	ordered set (increasing order) of all values from the interval $[V_{min}, V_{max}]$ that occur at least once in R; $V = \{v_i \mid 1 \leq\ i \leq\ n\}$
F	set of corresponding frequencies (non-zero); $F = \{0 <\ f_n \leq\ \|R\| \mid 1 \leq\ i \leq\ n\}$ therefore we have $\|R\| = \sum_{i=1}^{n} f_i$
n	$n = \|V\| = \|F\|$ (Note: $n \leq\ N$)
R^S	encrypted and bucketized relation, on server
M	maximum number of buckets
Q	set of all "legal" range queries over R
q	a random range query drawn from Q; $q = [l, h]$ where $l \leq\ h$ and $h, l \in [V_{min}, V_{max}]$
Q'	set of all bucket-level queries
q'	random bucket-level query drawn from Q'; basically q' is a sequence of at least one and at most M bucket identifiers.
$T(q)$	translation function (on the client side) which, on input of $q \in\ Q$, returns $q' \in\ Q'$
R_q	set of tuples in R satisfying query q
$R^S_{q'}$	set of tuples in R^S satisfying query q'
W	query workload, induces probability dist on Q

Table 1: Notations for Buckets

Note that for an ordered domain with N distinct values, there are $N(N+1)/2$ possible range queries in the query set Q. Before presenting our algorithm, we would like to point out a couple of things regarding our query model (i.e. the various query distributions that we consider). The problem of histogram construction for summarizing large data, has similarities to the present problem. Optimal histogram algorithms either optimize their buckets i) independent of the workload, by just looking at the data distribution [31] or ii) with respect to a given workload [32, 33]. In the first approach, the query distribution is implicitly assumed to be uniform (i.e. all possible range queries are equiprobable). We address both the cases, where in the query distribution is one of the following:

1) Uniform: All queries are equi-probable. Therefore probability of any query is $= \frac{2}{N(N+1)}$.

2) Workload-induced: There is a probability distribution P induced over the set of possible queries Q, where the probability of a query q is given by the fraction of times it occurs in the workload W (W is a bag of queries from Q).

We analyze the case of uniform query-distribution in detail here. We omit the discussion on how the general distribution (workload induced) case can be tackled due to space restrictions. The interested reader can refer to [11] for the algorithm.

3.2 Uniform query distribution

The total number of false positives (TFP), where all queries are equiprobable can be expressed as:

$$\text{TFP} \;=\; \sum_{\forall q \in Q} (\; |R^S_{T(q)}| - |R_q| \;)$$

The average query precision (AQP) can be expressed as (see notation in table 1):

$$\text{AQP} \;=\; \frac{\sum_{q \in Q} |R_q|}{\sum_{q \in Q} |R^S_{T(q)}|} \;=\; 1 - \frac{\text{TFP}}{\sum_{q' \in Q'} |R^S_{q'}|}$$

where $q' = T(q)$.

Therefore minimizing the total number of false positives is equivalent to maximizing *average precision* of all queries.

As before, consider a single attribute of a relation from a totally ordered discrete domain, such as the set of non-negative integers. For a bucket B, there are $N_B = (H_B - L_B + 1)$ distinct values where L_B and H_B denote the low and high bucket boundary, respectively. Let V_B denote the set of all values falling in range B and let $F_B = \{f^B_1, \ldots, f^B_{N_B}\}$ denote the set of corresponding value frequencies. Recall that Q is the set of all range queries over the given attribute. We need to consider all queries that involve at least one value in B and compute the total overhead (false positives) as follows:

Let the set of all queries of size k be denoted by Q_k and $q_k = [l, h]$ denote a random query from Q_k where $h - l + 1 = k$. Then, the total number of queries from Q_k that overlap with one or more points in bucket B can be expressed as: $N_B + k - 1$. Of these, the number of queries that overlap with a single point v_i within the bucket is equal to k. The case for $k = 2$ is illustrated in figure 2. Therefore, for the remaining $N_B - 1$ queries, v_i contributes f_i false positives to the returned set (since the complete bucket needs to be returned). Therefore, for all $N_B + k - 1$ queries of size k that overlap with B, the total number of false positives returned can be written as:

$$\sum_{v_i \in B} (N_B - 1) * f_i = (N_B - 1) * \sum_{v_i \in B} f_i$$

$$= (N_B - 1) * F_B \approx N_B * F_B$$

where F_B is the total number of elements that fall in the bucket (i.e., the sum of the frequencies of the values that fall in B). We make the following important observation here:-

Observation 2 *For the uniform query distribution, the total number of false positives contributed by a bucket B, for set of all queries of size k, is independent of k. In effect the total number of false positives contributed by a bucket (over all query sizes) depends only on the width of the bucket (i.e. minimum and maximum values) and sum of their frequencies.*

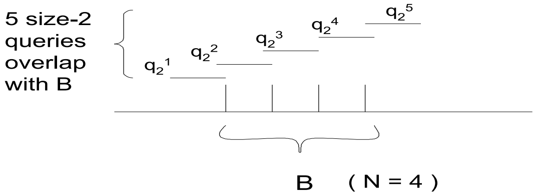


Figure 2: Queries overlapping with bucket

In light of the above observation, we conclude that minimizing the expression $N_B * F_B$ for all buckets would minimize the total number of false-positives for all values of k (the complete set of $\frac{N(N+1)}{2}$ range queries).

3.3 The Query-Optimal-Bucketization Algorithm (uniform distribution case)

As follows from the preceding discussion, our goal is to minimize the objective function : $\sum_{B_i} N_{B_i} * F_{B_i}$. Let $QOB(1, n, M)$ (*Query Optimal Bucketization*) refer to the problem of optimally bucketizing the set of values $V = \{v_1, \ldots, v_n\}$, using at most M buckets (Note that $v_1 < \ldots < v_n$, each occurring at least once in the table). We make the following two key observations:

1) Optimal substructure property: The problem has the optimal substructure property [34], therefore allowing one to express the optimum solution of the original problem as the combination of optimum solutions of two smaller sub-problems such that one contains the leftmost $M - 1$ buckets covering the $(n - i)$ smallest points from V and the other contains the extreme right single bucket covering the remaining largest i points from V:

$$\begin{aligned} QOB(1, n, M) \quad &= \quad Min_i[QOB(1, n-i, M-1) \\ &\qquad +BC(n-i+1, n)] \\ \text{where } BC(i, j) \quad &= \quad (v_j - v_i + 1) * \sum_{i \le t \le j} f_t \end{aligned}$$

($BC(i, j)$ is cost of a single bucket covering $[v_i, v_j]$)

2) Bucket boundary property: It can be intuitively seen that for an optimal solution, the bucket boundaries will always coincide with some value from the set V (i.e. values with non-zero frequency). Therefore in our solution space, we need to consider only buckets whose end points coincide with values in V, irrespective of the total size of the domain.

The algorithm solves the problem bottom-up by solving and storing solutions to the smaller sub-problems first and using their optimal solutions to solve the larger problems. All intermediate solutions are stored in the 2-dimensional matrix H. The rows of H are indexed from $1, \ldots, n$ denoting the number of leftmost values from V that are covered by the buckets for the given sub-problem and the columns are indexed by the number of maximum allowed buckets (from $1, \ldots, M$). Also note that the cost of any single bucket covering a consecutive set of values from

```
Algorithm: QOB$(D, M)$
Input: Data set $D = (V, F)$ and max # buckets $M$
      (where $|V| = |F| = n$)
Output: Cost of optimal bucketization & matrix $H$
Initialize
   (i) matrix $H[n][M]$ to 0
   (ii) matrix $OPP[n][M]$ to 0
   (iii) compute $EndSum(j) = EndSum(j+1) + f_j$
         for $j = 1 \ldots n$
For $k = 1 \ldots n$ // For sub-problems with max 2 buckets
   $H[k][2] = Min_{2 \leq i \leq k-1}(BC(1,i) + BC(i+1, K))$
   Store optimal-partition-point $i_{best}$ in $OPP[k][2]$
For $l = 3 \ldots M$ // For the max of 3 up to M buckets
   For $k = l \ldots n$
      $H[k][l] = Min_{l-1 \leq i \leq k-1}(H[i][l-1] + BC(i+1, k))$
      Store optimal-partition-point $i_{best}$ in $OPP[k][l]$
Output "Min Cost of Bucketization = $H[n][M]$"
end
```

Figure 3: Algorithm to compute query optimal buckets

V can be computed in constant time by storing the cumulative sum of frequencies from the right end of the domain, call them *EndSum* (i.e. $EndSum_n = f_n, EndSum_{n-1} = f_{n-1} + f_n \ldots$. Storing this information uses $O(n)$ space. We also store along with the optimum cost of a bucketization, the lower end point of its last bucket in the $n \times M$ matrix OPP (Optimal Partition Point) for each sub-problem solved. It is easy to see that the matrix OPP can be used to reconstruct the exact bucket boundaries of the optimal partition computed by the algorithm in $O(M)$ time. The dynamic programming algorithm is shown in figure 3[3] and an illustrative example is given below.

Example 2: Assume the input to QOB algorithm is the following set of (data-value, frequency) pairs: $D = \{(1,4),(2,4),(3,4),(4,10),(5,10),(6,4),(7,6),(8,2),(9,4),(10,2)\}$ and say the maximum number of buckets allowed is 4, then (figure 4) displays the optimal histogram that minimizes the cost function. The resulting partition is $\{1,2,3\},\{4,5\},\{6,7\},\{8,9,10\}$. Note that this histogram is not equidepth (i.e all bucket need not have the same number of elements). The minimum value of the cost function comes out to be $= 120$. In comparison the approximately equi-depth partition $\{1,2,3\},\{4\},\{5,6\},\{7,8,9,10\}$ has a cost $= 130$. $\diamondsuit$

3.3.1 Computation and Space Complexity

The complexity of the algorithm is $O(n^2 * M)$ which is dominated by the nested loop step, where the outer loop runs M times, inner loop runs $O(n)$ times and computing the minima over i takes another $O(n)$ computations. Computing cost of each bucket, the procedure $BC(i,j)$ can be done in $O(1)$ time if the sequence of numbers *EndSum* is precomputed, which again takes $O(n)$ time. The space complexity of the algorithm is clearly $O(n * M)$ due to the two matrices H and OPP. Due to lack of space, experimental observations for the running times of the algorithm is omitted in this paper and reported only in [11].

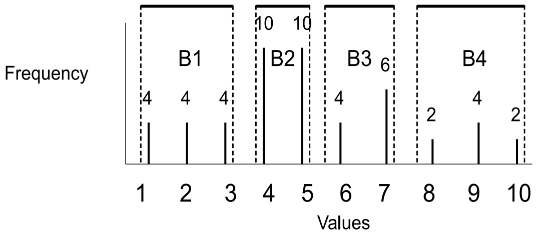


Figure 4: Optimum buckets for uniform query distribution

Before ending this section, we would like to point out that our techniques can be utilized to solve multi-attribute range queries as well. This can be done in a relatively straightforward manner, by utilizing the crypto-indices built on each of the query attributes and then returning the common set of tuples which satisfy the range constraints for each dimension. We present some more discussion on multi-attribute range queries in section 7.

3.4 Generic Query Workload

We briefly describe how the same dynamic programming algorithm of figure 3 can be used for an arbitrary distribution induced by a given query workload W. Here we will represent the workload $W = \{(q, w_q) | q \in W \cap w_q > 0\}$. Here w_q is the probability that a randomly selected query from W is same as q. It is easy to see that the same optimal substructure property, as presented above, holds independent of the probability distribution induced by W on the query space. The only difference is in computation of the bucket costs BC which translates into the computation of the array $Endsum[1, \ldots, n]$ in the preprocessing step. We present the algorithm to compute the entries of the *EndSum* array for a generic workload W in figure 5. Also note that now any bucket B covering range $[i,j]$ will have the cost given by $BC(i,j) = EndSum[i] - EndSum[j]$. Note that in case of a generic workload W, the preprocessing step takes $O(|W|log(n))$ time. The interested reader can refer to [11].

4 Privacy Measures

Here we propose two data-level privacy measures relevant to data bucketization and argue their utility.

4.1 Adversary's goal

The pillar of the DAS model is the untrusted server. In the context of our discussion, the adversary (denoted as A) collectively represents the server as well as any other malicious entities in the systems.

[3] in the workload-induced case, only the *EndSum* computation is done differently, the rest of the algorithms remains the same

```
Algorithm: Compute-EndSum($D, W, EndSum[1 \dots n]$)
Input: $D = (V, F)$, workload $W$ & $EndSum[1 \dots n]$
      (note: $EndSum[1 \dots n]$ is initialized to $0's$)
Output: Array $EndSum$ with entries filled in
For $i = (n-1) \dots 1$
  $EndSum[i] = EndSum[i+1]$
  For all $q = [l_q, h_q] \in W$ such that $h_q \in [v_i, v_{i+1})$
      $EndSum[i] = EndSum[i] + (\sum_{j=i+1}^{n} f_j) * w_q$
  For all $q = [l_q, h_q] \in W$ such that $l_q \in (v_i, v_n]$
      $EndSum[i] = EndSum[i] + f_i * w_q$
Output $EndSum[1 \dots n]$
end
```

Figure 5: EndSum for generic query workload W

While A's possible goals are difficult to enumerate, we focus on the context of the current application. In particular, we make the following two assumptions about A's goals:

Individual-centric information: A is interested in determining the precise values of sensitive attributes of some (all) individuals (records) with high degree of confidence. Eg: *What is the value of salary field for a specific record.* We refer to A's ability to estimate a value as **Value-Estimation-Power (VEP)** of A.

Query-centric information: A is interested in identifying the exact set of etuples that satisfy any (cleartext) query $q \in Q$ with the highest possible *precision* and *recall* [4]. Eg: *Which are the records of people who get salary between* $100K$ *and* $150K$. We denote A's ability to identify a set of etuples satisfying a query as **Set-Estimation-Power (SEP)**.

We point out an important distinction between the meaning of "precision" for A and for the data owner (client): We assume that the underlying row-level encryption employed by the client is to the table, is *unbreakable* (eg. some non-deterministic encryption algorithm). Therefore A never obtains the plaintext value of a sensitive attribute. Given an etuple, the best A can do is obtain a probabilistic estimate of the true value with high degree of confidence. Similarly, given a cleartext query q and a set S of bucketized etuples, A can assign a certain probability to whether any etuple in S satisfies q. On the other hand, there is no notion of uncertainty involved for the client. After receiving a set of etuples, he decrypts them and finds out exactly which tuples satisfy the query q. The overhead or imprecision for the client is only in terms of the extra false-positives that need to be decrypted and filtered out.

The above discussion makes it clear that A has to reconstruct the whole table by estimating/inferring the correct values of sensitive attributes. A can only achieve this by first *breaking* the bucketization scheme and using this knowledge to form *statistical estimates* of the set (range) of values that the attribute(s) of a tuple can take. We also assume that A employ statistical techniques that maximize its *degree of confidence* and minimize the cardinality (size) of set (range) of possible values (i.e., maximize his VEP and SEP).

To simplify the analysis, we make the following assumption, which allows us to perform the **worst-case** *breach of privacy* analysis:

Assumption 4.1 *A knows the entire bucketization scheme and the exact probability distribution of the values within each bucket.*

For example, given that bucket B has 10 elements, we assume A knows that: 3 of them have value 85, 3 have value 87 and 4 have value 95, say. However, since the elements within each bucket are indistinguishable, this does not allow A to map values to elements with absolute certainty. We now propose the two measures of privacy.

4.2 Variance

We propose the **Variance** of the distribution of values within a bucket B as its measure of "Individual-Centric-Privacy guarantee". We base our choice of variance on the theorem (see below), however, we first define the term *Average Squared Error of Estimation (ASEE)* as follows:

Definition 4.1 ASEE: *Assume a random variable X_B follows the same distribution as the elements of bucket B and let P_B denote its probability distribution. For the case of a discrete (continuous) random variable, we can derive the corresponding probability mass (density) function denoted by p_B. Then, the goal of the adversary is to* **estimate** *the true value of a random element chosen from this bucket. We assume that A employs a statistical estimator for this purpose which is, itself a random variable, X'_B with probability distribution P'_B.*

In other words, A guesses that the value of X'_B is x_i, with probability $p'_B(x_i)$. If there are N values in the domain of B, then we define **Average Squared Error of Estimation (ASEE)** *as:*

$$ASEE(X_B, X'_B) = \sum_{j=1}^{N} \sum_{i=1}^{N} p'_B(x_i) * p_B(x_j) * (x_i - x_j)^2$$

Theorem 4.2 $\mathbf{ASEE(X, X') = Var(X) + Var(X') +}$ $\mathbf{(E(X) - E(X'))^2}$ *where X and X' are random variables with probability mass (density) functions p and p', respectively. Also* $\mathbf{Var(X)}$ *and* $\mathbf{E(X)}$ *denote variance and expectation of X respectively.*

Proof: We have from definition 4.1, $ASEE(X, X')$ as

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} p'(x_i) p(x_j)(x_i - x_j)^2$$

$$= \sum_{i=1}^{N} p'(x_i) \sum_{j=1}^{N} p(x_j)(x_i - x_j)^2$$

[4] Precisions and recall refer to accuracy and completeness, respectively, of a set of etuples with respect to a given query

$$= \sum_{i=1}^{N} p'(x_i) \sum_{j=1}^{N} p(x_j)({x_i}^2 + {x_j}^2 - 2x_i x_j)$$

$$= \sum_{i=1}^{N} p'(x_i)[\sum_{j=1}^{N} p(x_j){x_i}^2 + \sum_{j=1}^{N} p(x_j){x_j}^2 - 2\sum_{j=1}^{N} p(x_j)x_i x_j]$$

Using $Var(X) = \sigma^2 = E(X^2) - \mu^2$ we get

$$= \sum_{i=1}^{N} p'(x_i)[1.{x_i}^2 + (\sigma^2 + \mu^2) - 2\mu x_i]$$

$$= \sum_{i=1}^{N} p'(x_i){x_i}^2 + (\sigma^2 + \mu^2)\sum_{i=1}^{N} p'(x_i) - 2\mu \sum_{i=1}^{N} p'(x_i)x_i$$

$$= ({\sigma'}^2 + {\mu'}^2) + (\sigma^2 + \mu^2) - 2\mu\mu'$$

$$= \sigma^2 + {\sigma'}^2 + (\mu - \mu')^2$$

$$= Var(X) + Var(X') + (E(X) - E(X'))^2$$

It is easy to see that A can minimize $ASEE(X_B, X'_B)$ for a bucket B in two ways: 1) by reducing $Var(X'_B)$ and 2) by reducing the absolute value of the difference $E(X_B) - E(X'_B)$. Therefore, the best estimator of the value of an element from bucket B that A can get, is the constant estimator equal to the mean of the distribution of the elements in B (i.e., $E(X_B)$). For the *constant estimator* X'_B, $Var(X'_B) = 0$. Also, as follows from basic sampling theory, the "mean value of the sample-means is a good estimator of the population (true) mean". Thus, A can minimize the last term in the above expression by drawing increasing number of samples or, equivalently, obtaining a large sample of *plaintext* values from B. However, note that the one factor that A cannot control (irrespective of the estimator he uses) is the true variance of the bucket values, $Var(X_B)$. Therefore, even in the best case scenario (i.e., $E(X'_B) = E(X_B)$ and $Var(X'_B) = 0$), A still cannot reduce the ASEE below $Var(X_B)$, which, therefore, forms the lower bound of the accuracy achievable by A. Hence, we conclude that the data owner (client) should try to bucketize data in order to **maximize the variance** of the distribution of values within each bucket.

4.3 Entropy

A higher value of variance tends to increase the average error of estimation for the adversary therefore increasing the individual centric privacy guarantee. Nonetheless, variance does not seem to be the appropriate measure of "query centric privacy guarantee" of a bucket. We then turn for help to information theory. As noted above, A's knowledge of the bucket contents are limited by the probability distribution, in the worst case (best case for the A, when he has learnt the complete bucketization). It is well-known that **entropy** of a random variable X is a measure of its uncertainty [17]. Entropy of a random variable X taking values $x_i = 1, \ldots, n$ with corresponding probabilities $p_i, i = 1, \ldots, n$ is given by:

$$Entropy(X) = H(X) = -\sum_{i=1}^{n} p_i \times log_2(p_i)$$

We propose that the measure of "query-centric privacy guarantee" given by a bucket B be the **entropy** of B's probability distribution (i.e., the distribution of values within B, as known to A, which happens to be the true distribution in the worst case scenario). We argue that entropy is an appropriate measure of query-centric privacy, by providing a simple example below. In the same example, we also show that variance and entropy are un-related, i.e., they are important and independent measures of privacy. A more formal argument for choosing entropy as the measure for query-centric privacy is provided in [11].

Example 3: Consider the following data set:
$T = \{(2,1),(4,1),(6,1),(8,1)\}$ where each value occurs with frequency 1.

A) There are 10 distinct range queries (classified by their solution sets) possible on this attribute. Consider the case when there are 4 buckets, each containing a single value. The bucket entropy is 0 if A knows the contents of the bucket, since A can identify the exact set of solution tuples for each of the 10 queries. Now consider only 2 buckets, e.g., as in row 1 of table 2. Each bucket in this case have entropy = 1. A can retrieve the precise set of tuples for only 3 of the 10 queries. For the remaining 7 queries, A can only specify the solution set with some probability. Therefore, we make the following observation:

Observation 3 *Increasing bucket entropy reduces the adversary's ability to identify tuples satisfying a query.*

B) To distribute the elements of T into two non-empty buckets, we have the following distinct partitions to consider (where σ^2 denotes variance and H denotes entropy for the respective buckets):

Partition	$\sigma^2{}_{\mathbf{B_1}}$	$\sigma^2{}_{\mathbf{B_2}}$	$\mathbf{H_{B_1}}$	$\mathbf{H_{B_2}}$
1) $B_1 = \{2,4\}; B_2 = \{6,8\}$	1	1	1	1
2) $B_1 = \{2,6\}; B_2 = \{4,8\}$	4	4	1	1
3) $B_1 = \{2,8\}; B_2 = \{4,6\}$	9	1	1	1
4) $B_1 = \{4,6,8\}; B_2 = \{2\}$	2.67	0	1.585	0
5) $B_1 = \{2,6,8\}; B_2 = \{4\}$	6.22	0	1.585	0
6) $B_1 = \{2,4,8\}; B_2 = \{6\}$	6.22	0	1.585	0
7) $B_1 = \{2,4,6\}; B_2 = \{8\}$	2.67	0	1.585	0

Table 2: Variance and Entropy of buckets

Obviously, case 2 represents the most desirable bucketization since it seems to balance variance as well as entropy. We note that the entropies of the buckets do not seem to be correlated with the respective variances.$\diamondsuit$

Therefore, as suggested earlier, we treat variance and entropy of each bucket, as two independent measures of privacy and try to achieve a partition that maximizes both simultaneously for every bucket.

4.3.1 Why Entropy

Here we show that entropy is an appropriate measure of query-centric privacy, especially with respect to range queries. We will deal with discrete data to keep our analysis simple. Also we will perform the analysis under the worst case scenario, where the adversary has **complete** knowledge of the data distribution within each bucket. Therefore in the discrete data case, the adversary is assumed to know the exact values and their frequencies within each bucket. We concretize the adversarial model below.

Definition 4.1 Adversarial Learning Model: *The domain is that of discrete values (could be integers or categories). Under the worst case adversarial model, A is assumed to know the exact frequencies of each of these distinct* **classes** *of data, within each bucket. But A does not know which element corresponds to which value due to their encrypted representations. We can consider each data value to be a distinct class in itself. We then assume that the goal of A is to* guess[5] *the true class of each element in a bucket. Therefore in our model, A assigns a class-label to each data element in a bucket, using only the knowledge of their frequencies (prior knowledge) and nothing else. The constraint being that the final labelling of the elements should be consistent with the frequencies of the values in the bucket. We call such assignments,* random consistent assignments (RCA's)*. The quality of such a RCA (or random classification if you may call it) is given by the* **proportion of correct class-assignments** *that are made. This defines a notion of "approximate correctness" of query solutions. We are interested in estimating the* average quality of solutions *that A can achieve for any range query, for a given RCA's. We will assume that all RCA's are equiprobable. For a given RCA, the adversary will retrieve the elements by looking at their labels.*

Now in the light of the above mentioned adversarial model, we need to show how the adversary's average quality of answers can be minimized. In other words, over all RCA's the average number of false positives occuring in a solution set of a range query, is maximized.

Now assume that there are M buckets $B_1, \ldots, B_M$ in all. Also let $q(D)$ denote the *query set* (i.e. set of records satisfying the query q from database D). For a given query q assume that there are n_1 elements in $B_1 \cap q(D)$, n_2 elements in $B_2 \cap q(D)$ and so on. Then in a random RCA (in B_1 say), the probability that any element in the bucket get assigned its correct class label $= \sum_{i \in B_1} p_i^2$, where i is a class in B_1 (i.e. distinct value) and p_i is the proportion of elements in this bucket from class i. Therefore for a query q, the number of correct records selected from B_1 for an RCA in B_1, will be $n_1 \times \sum_{i \in B_1} p_i^2$. The quality of a query-set for the adversary can be defined as follows:

$$Quality(q(D)) = \frac{n_1 \sum_{i \in B_1} p_i^2 + \ldots + n_M \sum_{i \in B_M} p_i^2}{|q(D)|}$$

In a RCA each element of a bucket is independently assigned a class label i with probability p_i (= proportion of class i within the bucket), therefore the probability of correct class assignment for an element $x \in B = \sum_{i \in B} P(x \in i, x \leftarrow i) = \sum_{i \in B} p_i^2$ where i varies over all the classes.

Our aim is to minimize the quality of query-sets for the adversary. Assuming that all the $O(n^2)$ entries are equiprobable, the coefficients $n_1, \ldots, n_M$ can be neglected since the bucket sizes are approximately the same (after diffusion) and therefore over all queries q, each bucket would roughly contribute the same number of records to the query set. Now our measure of quality becomes proportional to simply the factor $\sum_{i \in B} p_i^2$ for every bucket B. This as we know is a decreasing function of number of distinct classes in B and has a **minima** when all p_i's are equal. (Note $\sum_{i \in B} p_i = 1$). **Entropy** $(= - \sum_{i \in B} p_i log p_i)$ of the buckets is also a function with exactly similar characteristics and since entropy is the widely used measure of uncertainty in a distribution, we propose it to be the inverse measure of the quality of queries that an adversary can expect to derive. That is, increase in entropy of buckets signifies decrease in utility of the bucket for A.

(NOTE: This could be seen as a primitive kind of encoding of messages (here numeric values) that are sent over a channel by the source, to the user. But the channel is snooped on by an adversary who wants to determine the true value of an encoded value. We assume that he has knowledge of the encodings, (similar to buckets). The overhead is in terms of the fact that the source will have to send all elements having the same encoding, since he does not want the adversary, who might know the actual query as well, to determine the elements exactly corresponding to the given query.)

5 The Privacy-Performance Trade-off

This section studies the privacy-performance trade-off. Our goal is to develop a bucketization strategy that allows for this exploration to be carried out in a controlled manner. Section 3 gave the QOB-algorithm that computes, for a given number of buckets, the optimum bucketization of data leading to best performance (i.e. minimize false positives). Of course the optimal buckets also offer some level of privacy, but in many cases that might not be good enough (that is buckets might not have a large enough variance and/or entropy). What we now explore is how to re-bucketize the data, starting with the optimal buckets and allowing a bounded amount of performance degradation, in order to maximize the two privacy measures (**entropy** and

[5] since we consider single attributes, there is no other information that the adversary can use to determine the actual value of the elements

variance) simultaneously. We formalize the problem being addressed below:

Trade-off Problem: Given a dataset $D = (V, F)$ and an optimal set of M buckets on the data $\{B_1, B_2, \ldots, B_M\}$, re-bucketize the data into M new buckets, $\{CB_1, CB_2, \ldots, CB_M\}$ such that no more than a factor K of performance degradation is introduced and the **minimum variance** and **minimum entropy** amongst the M random variables $X_1, \ldots, X_M$ are simultaneously **maximized**, where the random variable X_i follows the distribution of values within the i^{th} bucket.

Solution approach: The above mentioned problem can be viewed as a multi-objective constrained optimization problem [19], where the entities *minimum entropy* and *minimum variance* amongst the set of buckets are the two objective functions and the constraint is the *maximum allowed performance degradation factor* K (we will call it the *Quality of Service* or the QoS constraint). Such problems are combinatorial in nature and the most popular solution techniques seem to revolve around the *Genetic Algorithm* (GA) framework [20], [21]. GA's are iterative algorithms and cannot guarantee termination in polynomial time. Further their efficiency degrades rapidly with the increasing size of the data set. Therefore instead of trying to attain optimality at the cost of efficiency, we design a novel algorithm which we call the *controlled diffusion algorithm* (CDf-algorithm). The CDf-algorithm increases the privacy of buckets substantially while ensuring that the performance constraint is not violated.

5.1 Controlled Diffusion

We can compute the optimal bucketization for a given data set using the QOB-algorithm presented in figure 3 of section 3, let us call the resulting optimal buckets $B_i's$ for $i = 1, \ldots, M$. The controlled diffusion process creates a new set of M approximately equidepth buckets which we call *composite buckets* (denoted by $CB_j, j = 1, \ldots, M$) by *diffusing* (i.e. re-distributing) elements from the B_i's into the CB_j's. The diffusion process is carried out in a controlled manner by restricting the number of distinct CB's that the elements from a particular B_i get diffused into. This resulting set of composite buckets, the $\{CB_1, \ldots, CB_M\}$ form the final bucketized representation of the client data.

The M composite buckets need to be approximately equal in size in order to ensure the QoS constraint, as will become clear below. The equidepth constraint sets the target size of each CB to be a constant $= f_{CB} = |D|/M$ where $|D|$ is size of the dataset (i.e. rows in the table). (We do not implement the equidepth constraint rigidly but as our experiments demonstrate, the error is still quite small). Let us see how the QoS constraint is actually enforced: If the maximum allowed performance degradation $= K$, then for an optimal bucket B_i of size $|B_i|$, we ensure that its elements are diffused into no more than $d_i = \frac{K * |B_i|}{f_{CB}}$ composite buckets (as mentioned above $f_{CB} = |D|/M$). We round-off the diffusion factor d_i to the closest integer. Assume that in response to a range query q, the server using the set of optimal buckets $\{B_1, \ldots, B_M\}$, retrieves a total of t buckets containing T elements in all. Then in response to the same query q our scheme guarantees that the server would extract no more than $K * T$ elements at most, using the set $\{CB_1, \ldots, CB_M\}$ instead of $\{B_1, \ldots, B_M\}$. For example, if the optimal buckets retrieved in response to a query q were B_1 and B_2 (here $t = 2$ and $T = |B_1| + |B_2|$), then to evaluate q using the CB_j's, the server won't retrieve any more than $K * |B_1| + K * |B_2|$ elements, hence ensuring that precision of the retrieved set does not reduce by a factor greater than K. An added advantage of the diffusion method lies in the fact that it guarantees the QoS lower bound is met not just for the average precision of queries but for each and every individual query. The important point to note is that the domains of the composite buckets overlap where as in the case of the optimal buckets, they do not. Elements with the same value can end up going to multiple CB's as a result of this diffusion procedure. This is the key characteristic that allows us to tweak the privacy measure while being able to control the performance degradation, in other words this scheme lets us explore the "privacy-performance trade-off curve". The controlled diffusion algorithm is given in figure 6. Though our method does not provably maximize the privacy measures, it is found to perform very well in practice. We illustrate the diffusion process by an example below.

```
Algorithm : Controlled-Diffusion($D, M, K$)
Input : Data set $D = (V, F)$,
        $M$ = # of $CB's$ (usually same as # opt buckets)
        $K$ = maximum performance-degradation factor
Output : An $M$-Partition of the dataset (i.e. $M$ buckets)

Compute optimal buckets $\{B_i, \ldots, B_M\}$ using $QOB$ algo
Initialize $M$ empty composite buckets $CB_1 \ldots, CB_M$
For each $B_i$
  Select $d_i = \frac{K*|B_i|}{f_{CB}}$ distinct $CB$'s randomly, $f_{CB} = \frac{|D|}{M}$
  Assign elements of $B_i$ equiprobably to the $d_i$ $CB's$
  (roughly $|B_i|/d_i$ elements of $B_i$ go into each $CB$)
end For
Return the set buckets $\{CB_j | j = 1, \ldots, M\}$.
end
```

Figure 6: Controlled diffusion algorithm

Example 4: Let us take the example 3.3 from section 3 and see how it works when we allow a performance degradation of up to 2 times the optimal ($K = 2$). Figure 7 illustrates the procedure. In the figure, the vertical arrows show which of the composite buckets, the elements of an optimal bucket gets assigned to (i.e. diffused to). The final resulting buckets are shown in the bottom right hand-side of the figure and we can see

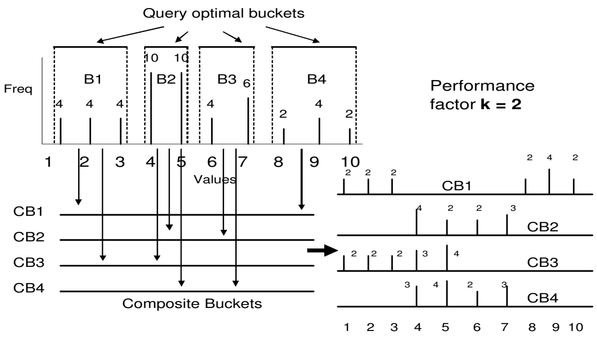


Figure 7: controlled diffusion (adhoc version)

that all the 4 CB's roughly have the same number size (between 11 and 14). The average entropy of a bucket increases from 1.264 to 2.052 and standard deviation increases from 0.628 to 1.875 as one goes from the B's to CB's. In this example the entropy increases since the number of distinct elements in the $CB's$ are more than those in the B's. The variance of the CB's is also higher on an average than that of the B's since the domain (or spread) of each bucket has increased. We can also guarantee that the average precision of the queries does not fall below a factor of 2 from the optimal, for instance take the range query $q = [2, 4]$, it would have retrieved the buckets B_1 and B_2 had we used the optimal buckets resulting in a precision of $18/32 = 0.5625$. Now evaluating the same query using the composite buckets, we would end up retrieving all the buckets CB_1 through CB_4 with the reduced precision as $18/50 \approx 0.36 > \frac{1}{2} * 0.5625$. (Note: Due to the small error margin allowed in the size of the composite buckets (i.e. they need not be exactly equal in size), the precision of few of the queries might reduce by a factor slightly greater than K). $\diamondsuit$

In the next section we address the case of multiple attribute range queries and then go on to discuss our experimental results in the subsequent section.

6 Expected Degradation in Query Precision

In this section we compute the bounds on the average factor of degradation in query precision due to the diffusion algorithm. We analyze the simple case when the initial buckets (optimal buckets) are actually equidepth (this can be the case when the data distribution is uniform) and show that it grows only logarithmically in the number of buckets used and is only slightly affected by the diffusion factor used (atleast when the optimal buckets correspond to equidepth buckets). This is good news since then we can increase the diffusion factor significantly without really paying too heavily in terms of false positives. In many cases, the equidepth partitioning itself gives a pretty good average case precision for all range queries. So the following analysis derives this relatively relaxed bound on the expected decrease- in-precision for range queries.

Original buckets as well as diffused buckets have the same size $\mathbf{f_{CB}} = \frac{\mathbf{N}}{\mathbf{M}}$ Let maximum allowed degradation factor be $\mathbf{d}$. Therefore each original bucket gets diffused into d composite buckets. Now we have M CB-labels to choose from for each original bucket and for diffusion factor d, each of the M labels gets selected with probability $\frac{d}{M}$. Therefore expected number of intersections for the image set of two original buckets is $= \frac{Md^2}{M^2} = \frac{d^2}{M}$. Similarly for three buckets the expected size of their intersections $= \frac{d^3}{M^2}$ and so on.

Useful result on counting: The number of elements in the union of n sets is

$$|\cup_{i=1}^{n} X_i| = \sum_{i=1}^{n} |X_i| - \sum_{i<j} |(X_i \cap X_j)| + \sum_{i<j<k} |(X_i \cap X_j \cap X_k)| \ldots$$

Now we compute the average by categorizing the queries into classes by the number of distinct original buckets they span (these buckets are contiguous) and using the linearity of expectations property. 1) For queries in Q_1 which span one bucket, number of records retrieved will be $E_1 = df_{CB} = \frac{dN}{M}$
2) For queries in Q_2, # records retrieved

$$E_2 = df_{CB} + df_{CB} - \frac{d^2 f_{CB}}{M} = f_{CB}(2d - \frac{d^2}{M}) = N[\frac{2d}{M} - \frac{d^2}{M^2}]$$

3) For queries in Q_3, the expected size of the set of records returned for a query, will be given by

$$E_3 = 3df_{CB} - \frac{3d^2 f_{CB}}{M} + \frac{d^3 f_{CB}}{M^2} = f_{CB}(3d - \frac{3d^2}{M} + \frac{d^3}{M^2})$$

Taking d out and using $f_{CB} = N/M$, we have

$$E_3 = N[\frac{3d}{M} - \frac{3d^2}{M^2} + \frac{d^3}{M^3}] = N - N[1 - \frac{3d}{M} + \frac{3d^2}{M^2} - \frac{d^3}{M^3}]$$

Now substituting $v = \frac{d}{M}$, we have $E_3 = N - N(1-v)^3$. Therefore for a general k, we get $E_k = N - N(1-v)^k$. So now, we compute the average factor of **increase** in the size of the solution set returned for range queries as:
Let $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_M$, then the average increase can be written as

$$E(increase) = \frac{1}{|Q|}[\frac{|Q_1|Nv}{\frac{N}{M}} + \frac{|Q_2|(N - N(1-v)^2)}{\frac{2N}{M}}$$

$$+ \frac{|Q_3|(N - N(1-v)^3)}{\frac{3N}{M}} + \ldots + \frac{|Q_M|(N - N(1-v)^M)}{\frac{MN}{M}}]$$

Now observe that $|Q_i| \propto (M - i + 1)$, when we have equidepth buckets to start with. And $|Q| \propto \frac{M(M+1)}{2}$, using these, we have

$$E(factor) = \frac{2}{N(M+1)}[MNv + \frac{M-1}{2}(N - N(1-v)^2) +$$

$$\frac{M-2}{3}(N-N(1-v)^3)+\ldots+\frac{M-(M-1)}{M}(N-N(1-v)^M)]$$

Now put $u = 1 - v$ and we have

$$E(factor) = \frac{2}{N(M+1)}[MN(1-u)+\frac{M-1}{2}(N-Nu^2)+$$

$$\frac{M-2}{3}(N-Nu^3)+\ldots+\frac{1}{M}(N-Nu^M)]$$

Cancelling out the factor N

$$E(factor) = \frac{2}{M+1}[M(1-u)+\frac{M-1}{2}(1-u^2)+$$

$$\frac{M-2}{3}(1-u^3)+\ldots+\frac{1}{M}(1-u^M)]$$

$$E(factor) = \frac{2}{M+1}[M(1-u)+\frac{M-1}{2}(1-u^2)+$$

$$\frac{M-2}{3}(1-u^3)+\ldots+\frac{1}{M}(1-u^M)]$$

$$E(factor) = \frac{2}{M+1}[(M+\frac{M-1}{2}+\frac{M-2}{3}+\ldots+\frac{1}{M})$$

$$-(Mu+\frac{M-1}{2}u^2+\frac{M-2}{3}u^3+\ldots+\frac{1}{M}u^M)]$$

Of course $u < 1$. Now one way of bounding the quantity $E(factor)$ from above, is to upper bound the first term (inside the square brackets) and lower bound the second term then the difference of these two will give us an upper bound. Let us call these A and B. So A can be upper bounded by rewriting as follows:

$$A = (M+\frac{M}{2}+\frac{M}{3}+\ldots+\frac{M}{M})$$

$$-(0+\frac{1}{2}+\frac{2}{3}+\ldots+\frac{M-1}{M})$$

We lower bound the second term in A by replacing all the non-zero fractions by $\frac{1}{2}$. Then $A \leq MH_M - \frac{M-1}{2}$, where H_M is the harmonic sum of the first M terms. Similiarly B can be bounded from below by replacing all the coeffients by $\frac{1}{M}$ as follows:

$$B = Mu+\frac{M-1}{2}u^2+\frac{M-2}{3}u^3+\ldots+\frac{1}{M}u^M$$

$$\geq \frac{1}{M}(u+u^2+u^3+\ldots+u^M)$$

$$\geq \frac{1}{M}u\frac{u^M-1}{u-1}$$

Therefore the $E(factor)$ can be bounded from above as follows:

$$E(factor) \leq \frac{2}{M+1}[MH_M-\frac{M-1}{2}-\frac{u(u^M-1)}{M(u-1)}]$$

Now substituing back for $u = \frac{M-d}{M}$, we get

$$E(factor) \leq \frac{2}{M+1}[MH_M-\frac{M-1}{2}-\frac{\frac{M-d}{M}[(\frac{M-d}{M})^M-1]}{M(\frac{M-d}{M}-1)}]$$

$$= \frac{2}{M+1}[MH_M-\frac{M-1}{2}+\frac{M-d}{dM}[(\frac{M-d}{M})^M-1]]$$

$$= \frac{2}{M+1}[MH_M-\frac{M-1}{2}-\frac{M-d}{dM}[1-(\frac{M-d}{M})^M]]$$

Put $t = \frac{d}{M}, whered \leq M, d \geq 1$

$$E(factor) \leq \frac{2}{M+1}[MH_M-\frac{M-1}{2}-\frac{1-t}{tM}[1-(1-t)^M]]$$

$$\leq 2H_M - \{O(1)+\frac{2(1-t)}{tM(M+1)}[1-(1-t)^M]\}$$

Now the sum of the terms in the parantheses is $O(1)$ and using the fact $H_M = ln(M) + \gamma$ (where $\gamma \approx 1.14$), we can bound the $E(factor)$ term from above by:

$$\mathbf{E(factor) \leq 2ln(M) + 2\gamma}$$

Hence showing that the factor of reduction of drop in precision for the range queries (atleast for the equidepth bucket case which corresponds to the optimal bucketization in case the data distribution is uniform) is bounded above by a small factor of natural logarithm of the number of buckets used.

7 Multi-Attribute Range Queries

We have so far discussed the privacy performance trade-off in range queries in context of a single partitionable attribute. In reality, range queries might refer more than one attribute. A straightforward approach is to apply the privacy enhanced bucketization strategy proposed earlier to each attribute individually. But in multi-attribute case another problem arises: that of *exposure via associations.* In such cases, the unique combination of bucket-tags corresponding to the different attributes in a single tuple might be used to disclose the identity of the owner or perhaps narrow down the space of possible values for critical fields. We first address the issue of *identity disclosure* of the owner of a tuple.

In multi-attribute datasets k-anonymity has been proposed as a measure of privacy [14]. k-anonymity is defined as follows:

Definition 7.1 *k-anonymity is said to hold, when encoding of attribute values in a table are such that for any row r, we can find at least $k-1$ other rows with the same encodings of the corresponding columns. That is for any row, there are at least $k-1$ other indistinguishable rows.*

A goal of adversary A might be to identify the record corresponding to a certain individual I. If A is able to learn the encodings for a few of the attributes for $I's$ record (i.e. the bucket-tags) such that these encoding, together distinguish $I's$ row from any other row, then disclosure is said to have occurred. Therefore it becomes critical to ensure a minimum k level of anonymity for any row of the given table where k is the desired privacy measure. Obviously a higher level of anonymity will reduce the chances of disclosure of any record of interest.

Ensuring k-anonymity in multi-attribute tables can generally tend to get more difficult with the increasing number of attributes that have to be indexed and it has been shown to be NP-Hard in [15]. For tables where only a single attribute needs to be indexed, the *level* of anonymity achieved for any tuple is of course the size of the bucket it is assigned to. The case where multiple attributes have to be indexed, is a more complicated one. Let us consider the special case where there is no correlation between the values of the various attributes and let $N_{A_1B_1}$ tuples have bucket-id B_1 for the attribute A_1. Then we might expect **on an average** to be able to find $P-1$ other tuples for a given tuple that are indistinguishable from it, where $P = \frac{N_{A_1B_1}}{k_2 * k_3 * \ldots * k_M}$ and attribute A_2 is partitioned into k_2 buckets, A_3 into k_3 buckets and so on to all M attributes. That is we can expect P-anonymity in general where P is defined as above. The value of P rapidly decreases with both, the increasing number of indexed attributes as well as the number of buckets allowed for each of these attribute domains.

Another attack that is a possibility in presence of multiple crypto-indices, is that of *prediction through association.* For instance when the adversary A knows about correlations between different attributes A_i and A_j say, he might be able to predict with high probability the value of attribute A_j of a tuple if he knows the bucketization of the A_i field. Such attack scenarios have been analyzed previously in [3].

Recently authors in [3] have explored the issue of exposure and quantified it in a different setting where hashing has been used instead of bucketization. Based on their work, one could possibly develop the framework to explore exposure in the bucketization case as well. However, a novel strategy proposed in [8] overcomes this problem in a different manner, instead of bucketizing attributes individually, it is done as a multidimensional partitioning. The authors show that the problem of exposure disappears when multidimensional partitioning is used. We feel our proposed diffusion based approach can be adapted to work in the case of multi-dimensional partitioning as well. But any such discussion is out of scope of this paper and will addressed in future work.

8 Experiments

We start by introducing the datasets we used and our experimental setup.

8.1 Datasets and experimental setup

The following two dataset and query set were used:
1)*Synthetic Data Set*: consists of 10^5 integer values generated uniformly at random from the domain $[0, 999]$.
2)*Real Data Set*: consisted of 10^4 data points taken from one of the columns of the “Co-occurrence Texture” table of the “Corel Image” dataset in UCI-KDD archive [18]. The readings correspond to the *angular momentum component* of some colored images. The values came from a real domain (roughly $(-0.800000, 8.000000)$). The frequency of most values was equal to 1 (i.e. unique) or some small integer c.
3)*Benchmark Query Set*: We generated two different set of queries corresponding to the synthetic and real datasets, Q_{syn} and Q_{real} respectively. Each were of size 10000 and were generated uniformly at random from the same ranges as the datasets themselves[6].

We carried out all our experiments on a 1G Hz pentium machine, with 512MB RAM.

8.2 Experiments

We carried out four sets of experiments that measured the following:
1) Decrease in Precision: of evaluating the benchmark queries using optimal buckets (QOB-buckets) and composite buckets (CB's). Figure 8 (a) plots the ratio of the **average precision** using optimal buckets to that using the composite buckets for the benchmark query set Q_{syn} as a function of **# of buckets**. Plots are shown for multiple values of the maximum allowed *performance degradation factor* $K = 2, 4, 6, 8, 10$. Figure 8 (b) shows the corresponding plot for the real dataset on query set Q_{real}.
2) Privacy Measure: We plot the ratio of **average standard deviation** of the CB's to that of the QOB-buckets as a function of **# of buckets** in figure 9 (a) for the synthetic dataset and in figure 9 (b) for the real dataset (for values of $K = 2, 4, 6, 8, 10$). Figure 10 (a) plots the ratio of **average entropy** for the two sets of buckets, on the synthetic dataset and figures 10 (b) display the corresponding ratio for the real dataset.
3) Performance-Privacy trade-off: Figure 11 (a) displays the trade-off between **average standard deviation** of buckets and **average precision**. We fix 6 values of M, the # of buckets $M = 100, 150, \ldots, 350)$ and for each M, we plot the average standard deviation of the optimal set of buckets (QOB's) as well as the average standard deviation for the 5 sets of composite buckets (CB's) obtained from the QOB's by applying the controlled-diffusion algorithm by setting the

[6] though real life datasets and queries rarely come from an uniform distribution, we feel the results reported here still give a good indication of the usefulness of our algorithms

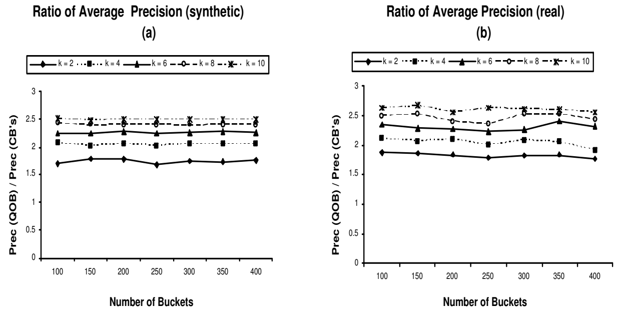


Figure 8: Decrease in Precision a)Synthetic b) Real data

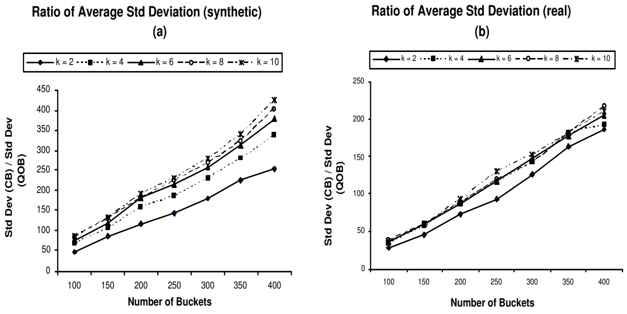


Figure 9: Increase in Std Dev a)Synthetic b) Real data

degradation factor $K = 2, 4, 6, 8, 10$. In effect we plot 6 different points for each value of M. Similar plots of **entropy-precision trade-off** for the same sets of buckets are plotted in figure 11 (b).

4) Time taken: to compute the optimal buckets by the QOB validates the $O(N^2M)$ complexity of the QOB algorithm. But the figures are excluded from this paper due to lack of space. The interested reader can refer to [11] for the plots.

8.3 Results

The empirical results on both datasets for the benchmark queries are quite positive in most of our experimental runs: the relative precision measurements for instance show that for most cases, the degradation in average precision is actually much smaller than the allowed maximum of K. In both the datasets, even when the maximum degradation factor allowed is 10, the observed decrease in query precision was less than 3. This is obviously because the controlled-diffusion algorithm leads to some degree of overlap in $range$[7]. This results in a much smaller drop in the precision than K, for a majority of the range queries.

Amongst the privacy measures, standard deviation increases by a large factor in most cases even for a small value of K. Whereas entropy, being a logarithmic measure, grows more slowly. Though for some values of K for the real data, average entropy of buckets decreased slightly as a result of diffusion.

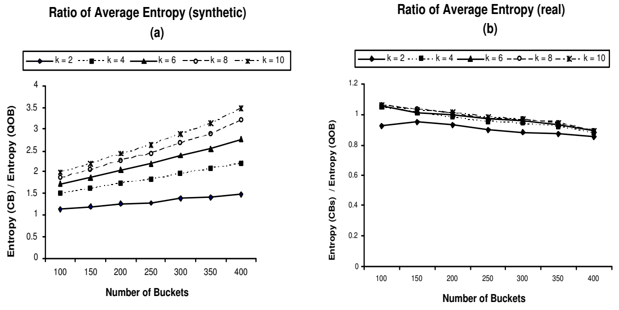


Figure 10: Change in Entropy a)Synthetic b) Real data

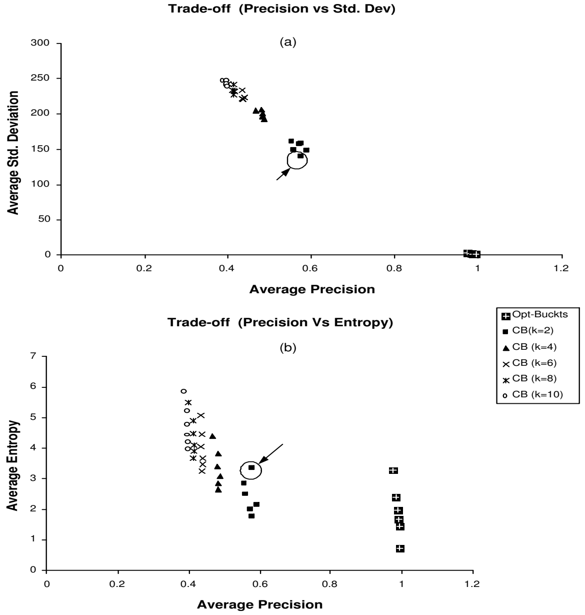


Figure 11: Privacy-Performance trade-off a)Std Dev vs Precision b) Entropy vs Precision

Finally the plots in the performance-privacy trade-off space display all the **design points** available to the data-owner. These plots provide a good estimate of the degree of privacy available if one is willing to sacrifice efficiency by a given amount. For e.g, by looking at the plots of figures 11 (a) and 11(b), the data owner might choose a bucketizaton scheme that uses 100 buckets and sets $K = 2$ since it provides a high value for average entropy as well as a sufficiently high value of standard deviation of the buckets for a small loss in efficiency (the design-point is circled in both the figures). One can also use non-integral values of K and explore more points in the trade-off space that might meet one's requirements). The trade-off plots could be made more accurate by choosing an appropriate set of benchmark queries for a given application.

[7] where $range(B)$ is the set of CB's that the elements of B are diffused into

9 Related Work

Privacy and security in databases have been a core area of research for a few decades now. Privacy related problems spring up in many sub-areas of database research, for e.g., access control [26], inference control [29],[28], statistical disclosure control [16], statistical databases [22], etc. More recently quite a bit of research has been done in areas such as privacy preserving data mining [9] [10], DAS oriented work (which we have reviewed earlier) and privacy-preserving information retrieval [30]. We give a brief summary of the research in few of the areas other than DAS.

Privacy Preserving Data mining: There does not seem to be any universally acceptable definition of privacy and the general trend is to define a notion of privacy that is best suited to the application at hand. Authors previously have suggested different measures for various application: Agrawal and Srikant [9] address the classification problem and propose the size l of the interval to which the value of a variable can be restricted with a confidence c to be the measure of privacy at c-confidence level. Agrawal and Aggarwal [10] study distribution reconstruction from randomized data for which they propose measures based on entropy and *mutual information* of random variables. Association rule mining in privacy-preserving manner has been addressed in [2]. [12, 13] take a cryptographic approach to compute decision trees and do EM-clustering for a distributed setting, in a privacy-preserving manner.

Statistical Database Protection and Disclosure Control: The central problem addressed in statistical databases and disclosure control is that of releasing datasets in a manner such that an individual interested in learning aggregate level measures (mean, median, frequency) is able to do so with minimum error and at the same time the data owner is able to secure the values of records from being *disclosed* [22], [23], [24], [25]. Data-perturbation by statistical noise addition is an important method of enhancing privacy [22], [24]. The idea is to perturb the true value by a small amount ϵ where ϵ is a random variable with a mean $= 0$ and a small variance $= \sigma^2$. Statistical disclosure control techniques consist of *generalization, suppression and data swapping* amongst others [16]. Another important privacy-measure, that of k-anonymity [14] has been introduced earlier in section 7. The literature in the area is vast and we refer the interested readers to [23] and [16] for a thorough survey of the field.

Inference Control: The inference problem in databases occurs when sensitive information can be disclosed from non-sensitive data and meta-data. A vast amount of research exists in this area as well [27]. An inference analysis based on variance of estimators has been carried out by the authors in [24] for randomization. Specifically of interest to us are the issues of inference control in i) general purpose databases, ii)statistical databases, iii)data mining and iv)web-based inferencing. Detection of inference channels in above scenarios is an important problem and has been addressed by many researchers. Due to lack of space, we refer the interested reader to [27],[28] and [29] for a detailed exposition and pointers to further literature in the area.

10 Conclusions

In this paper, we investigated "data bucketization" as a privacy-enhancing technique and highlighted the fundamental tradeoff between privacy and performance. We presented some inferencing and disclosure scenarios (that an adversary, primarily the untrusted server, might be interested in) and proposed two useful measures of privacy. We also derived an optimal algorithm for data partitioning that provably minimizes performance overhead in query processing. We presented the *controlled diffusion* algorithm that lets the data owner fine-tune bucketization to achieve the desired level of data privacy by sacrificing the accuracy (of query evaluation) by a small measured amount. The effectiveness of our proposed algorithms is validated by our experiments that show promising results on both synthetic and real datasets.

11 Future Work

In the future, we intend to assess the privacy loss in the case when the adversary has partial information about buckets (which is a more realistic scenario) instead of the worst case scenario where the adversary has complete information as considered here. Also, analyzing disclosure risk and privacy guarantee in case of multi-attribute data is an important goal of our research. Furthermore, we intend to explore/develop optimal partitioning algorithms that **provably maximize** privacy within performance constraints set by the data owner.

We recognize that many other research challenges would arise if we were to incorporate and deploy a bucketization strategy (such as the one proposed in this paper) in real systems. For example, in this paper we assumed a static database. Over time, as the database changes, the bucketization might not remain optimal and will have to be adapted. One approach is to construct a new bucketization periodically (on the fly) and replace the old index with a new one. To make the scheme more attractive for the on-line setting, it is interesting to explore incremental construction and migration schemes. Such incremental schemes have been developed for B-tree or other indices and some of these ideas might apply in our context as well.

Finally, we note that our focus has been on range queries and we showed how bucket content diffusion as a strategy can offer higher privacy with bounded overhead. Approaches involving the privacy/performance tradeoff in the context of other kinds of queries (such as join queries) remain part of our future work.

References

[1] Domingo-Ferrer, J., Castillo, R., X., S. An implementable scheme for Secure Delegation of Computing and Data. *ICICS, 1997*, pp. 445-451.

[2] Evfimievski, A., Srikant, R., Agrawal, A., Gehrke, J. Privacy reserving mining of association rules. SIGKDD, 2002.

[3] Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., Samarati, P. Balancing Confidentiality and efficiency in untrusted relational DBMSs. In*10th ACM CCS, 2003*

[4] Hacigumus, H., Iyer, B., Li, C., Mehrotra, S. Executing SQL over Encrypted Data in the Database Service Provider Model, *SIGMOD 2002*, June 4-6, Madison, Wisconsin, USA.

[5] Hacigumus, H., Iyer, B., Mehrotra, S. Efficient Execution of Aggregation Queries over Encrypted Relational Databases, In *DASFAA, 2004*, pp. 125-136.

[6] Bouganim, L., Pucheral, L. Chip-Secured Data Access: Confidential Data on Untrusted Servers, *In Proc. of the* 28^{th} *VLDB Conference, 2002.*

[7] Maheshwari, U., Vingralek, R., Shapiro, W. How to build a Trusted Database System on Unstrusted Storage *OSDI 2000*

[8] Jammalamadaka, R., Mehrotra, S. Querying Encrypted XML document *technical report TR-DB-04-03*, www-db.ics.uci.edu/pages/publications/index.shtml

[9] Agrawal, R., Srikant, R. Privacy-Preserving Data Mining. *ACM SIGMOD 2000*

[10] Agrawal, D., Aggarwal, C., C. On the Design and Quantification of Privacy Preserving Data Mining Algorithms. *PODS, 2001*

[11] Hore, B., Mehrotra, S., Tsudik, G. A Privacy-Preserving Index for Range Queries *technical report TR-DB-04-04*, www-db.ics.uci.edu/pages/publications/index.shtml

[12] Lindell, Y., Pinkas, B. Privacy Preserving Data mining. In Advances in Cryptology-CRYPTO 2000, pp. 36-54.

[13] Lin, x., Clifton, C. Distributed EM clustering without sharing local information. Journal of Information sciences, Feb 2003.

[14] Samarati, P., Sweeney, L. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical Report, SRI International 1998.

[15] Meyerson, A., Williams R. General k-anonymization is Hard. Tech-report CMU-CS-03-113.

[16] Willenborg, L., De Waal, T. Statistical Disclosure control in Practice. Springer-Verlag, 1996.

[17] Cover, T., M., Thomas, J, A. Elements of Information Theory. John Wiley & Sons, Inc., 1991.

[18] Corel Image Features database, UCI-KDD Archive.

[19] Steuer, R., E. Multiple Criteria Optimization - Theory, Computation and Application, Wiley, 1986.

[20] Goldberg D., E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts, 1988.

[21] Jones, D.R. and Beltramo, M.A. Solving Partitioning Problems with Genetic Algorithms, *Proc. of the 4th international conference on Genetic Algorithms*, 1991.

[22] Traub, J., F., Yemini, T., and Wozniakowski, H. The Statistical Security of a Statistical Database. TODS 1984, 672-679.

[23] Shoshani, A. Statistical Databases: Characteristics, Problems, and some Solutions. VLDB 1982, pp.208-222.

[24] Muralidhar, K., Sarathy, R. Security of Random Data Perturbation Methods. TODS 2000, 487-493.

[25] Yu, C., T., Chin, F., Y. A study in protection of statistical databases. SIGMOD 1977, pp. 169-181.

[26] Lunt, T. Access control policies for database systems. In Database Security II:Status and prospects, pp.41-52.

[27] Farkas, C., Jajodia, S. The Inference Problem: A Survey. SIGKDD Explorations, Newsletter, Vol 4, 6-11.

[28] Catalytic inference analysis:Detecting inference threats due to knowledge discovery. In the IEEE Symposium on security and Privacy, 1997, pp.188-199

[29] Thuraisingham, B. The use of conceptual structures for handling the inference problem. In Database Security V, pp.333-362

[30] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M. Private Information Retrieval. Proc of 36^{th} FOCS (1995), pp.41-50.

[31] Gilbert, A., C., Kotidis, Y., Muthukrishnan, S., Strauss, M.,J. Optimal and Approximate Computation of Summary Statistics for Range Aggregates *PODS, 2001* pp. 227-236.

[32] Gunopulos, D., Kollios, G., Tsotras, V., J. Approximating Multi-dimensional Aggregate Range Queries over Real Attributes *ACM-SIMOD, 2000*, pp. 463-474.

[33] Bruno, N., Chaudhuri, S., Gravano, L. STHoles: a multidimensional workload aware histogram *ACM SIGMOD 2001*, pp. 211-222.

[34] Cormen, T., H., Leiserson, C., E., Rivest, R., L. Introdcution to Algorithms, MIT Press.