

Stochastic Protection of Confidential Information in Databases: A Hybrid of Data Perturbation and Query Restriction

Manuel A. Nunez, Robert S. Garfinkel, and Ram D. Gopal

School of Business, University of Connecticut, 2100 Hillside Road, Storrs, Connecticut 06269, mnunez@business.uconn.edu, rgarfinkel@business.uconn.edu, ram@business.uconn.edu

Data perturbation and query restriction are two methods developed to protect confidential data in statistical databases. In the former the data is systematically changed to yield answers to queries that are statistically similar to those that would have resulted from the original data. The latter provides exact answers to queries as long as the risk of exact disclosure of confidential data does not become too great. We present a new methodology to combine these techniques so that the advantages of both are captured. The model is appropriate and computationally viable for large databases whether the queries are linear or nonlinear. The query restriction phase consists of finding an optimal subset of queries to answer exactly without compromising the database. This is an \mathcal{NP} -hard problem with a matroid intersection structure that lends itself to an efficient greedy heuristic. Then, given the queries that are answered exactly, we implement a data perturbation phase that provides stochastic protection and consistency. We present computational results on a large database with both linear and nonlinear queries. The results indicate that many queries can be answered exactly and the proposed perturbation approach provides more accurate answers than the standard perturbation method.

Subject classifications: statistical databases; database security; query restriction; data perturbation; matroid intersection; Hilbert spaces; nonlinear least squares estimation.

Area of review: Computing and Decision Technology.

History: Received February 2004; revisions received February 2005, December 2006, July 2006; accepted September 2006.

1. Introduction

With the proliferation of information technologies, both public and private organizations increasingly collect, house, process and disseminate information via database systems. A number of government census bureaus (e.g. Statistics New Zealand at www.stats.govt.nz) are releasing data over the Internet to enable users to effectively choose and customize the data that best suits their information needs. As the volume and scope of data, and usage of these systems continues to grow, there is increasing concern regarding the security of confidential information resident in them. As a result, the database manager (DBM) is charged with the twin responsibilities of providing users effective access to information, and ensuring the security of confidential information in the database.

A number of studies have addressed the issue of the conflict between these two responsibilities in the context of inference control mechanisms (see comprehensive surveys in Castano et al. 1996, Denning 1982), the objective of which is to provide answers to user queries in a manner that does not disclose individual confidential data. The threats to data security arise from a user's ability to infer confidential data from the query answers provided by the DBM. Most of the proposed methods can be classified as either query restriction or perturbation.

The query restriction methods that have been proposed have two distinctive properties. The first is the possibility of refusal of user queries, and the second is that query answers are exact. For any query, exact answers must always be the most valuable to the users of the database, and query restriction methods aim to provide these as often as possible. Two different classes of query restriction methods

have been developed in the literature. In the *design stage* class of query restriction methods, generally referred to as “cell suppression” (e.g. Cox 1980, Fischetti and Salazar 1999), the DBM pre-selects a set of queries for which exact answers will be provided. This set of queries is chosen such that their answers do not result in disclosure of any confidential numerical datum with certainty. Answers are typically released in the form of tables and no additional information on the confidential data is provided to the users.

On the other hand, the *run time stage* class of query restriction methods (see Chin and Ozsoyoglu 1982, Gopal et al. 1998) is designed for queries that arrive over time. Then a query will be declined if its answer, along with answers already provided to the user asking the query, is considered likely to result in disclosure of confidential data. The computational effort that must be paid for providing exact answers while maintaining security can be very high, thus the run time stage query restriction methods that have been proposed are suited to answer only linear and MIN or MAX queries. It is not, in general, a trivial problem to determine whether the incoming query would be too risky to answer even if only this limited class of queries is considered. Moreover that determination would differ among the various users so that the DBM would have to somehow keep track of the knowledge that each user has been afforded at any time. A further implication is that collusion among users is a serious problem, unlike in the design stage class of techniques, so that it is necessary to keep track of the information accumulated by each user from answered queries. This can be accomplished, for example, by maintaining a log of answered queries for each user (Chin and Ozsoyoglu 1982) or by bounding the representation of the answer space (Gopal et al. 1998). A final observation is that these methods operate in a local mode in the sense that decisions are made in real time, without considering the queries that may arrive in the future.

In contrast perturbation methods (see e.g. Duncan and Mukherjee 2000, Muralidhar et al. 1995, 1999, 2001, Traub et al. 1984) typically provide users with query answers from a database in which confidential data is perturbed. It follows that deterministic protection of confidential data is guaranteed, since the original confidential data can be discarded once the perturbation has been performed. Thus there is a *design stage*, in which the parameters of the perturbation are determined, followed by a run time stage in which answers based on the perturbed data can be given for any query type. The data is typically perturbed so that it satisfies probability distribution properties and is therefore designed to provide statistically close answers to queries. It follows that “acceptable” answers are likely to result for large cardinality queries, i.e. queries that refer to a large number of subjects of the database. On the other hand that may very well not be the case for small cardinality queries. These methods are simple and efficient to implement and do not require any type of recording of user knowledge. They are also able to provide measures of stochastic protection of the confidential data.

From the above discussion it can be seen that, in terms of upsides and downsides, perturbation can be considered to be almost a mirror image of run time stage query restriction. It has only two clear downsides, namely that query answers are neither exact nor deterministically correct and that the quality of a query answer can vary based on the query type and cardinality. In this work we consider ways to develop techniques that combine the upsides of design stage query restriction and perturbation. The motivations for such a technique are as follows. First, for any query, exact answers must always be the most valuable to the users of the database and therefore there is considerable value in providing these as often as possible. But, in run time query restriction there may be no way to model the problem of maximizing the value of exact answers since queries arrive over time and the DBM cannot precisely predict what they will be. Thus previous run time query restriction approaches have made these decisions on the fly, that is, either answer a query exactly or refuse it. Now, since completely refusing a query is not a friendly option one would like to at least give inexact answers to those queries that would otherwise have been refused. Therefore one is tempted to use a technique such as perturbation to answer the remainder of the queries. So, the idea of marrying an exact with an inexact technique is natural but has never been proposed in the literature.

The hybrid approach of providing exact answers to some queries and perturbed answers to others is already in use. For instance, Statistics New Zealand (2003) provides exact answers for MEAN and MEDIAN queries referring to more than six data points. Note that MEAN and MEDIAN queries are undoubtedly chosen for exact solution because these are deemed to be important and frequently asked. On the other hand, the lower limit of six for the number of data points is surely motivated by the inherent discomfort with giving exact solution to low cardinality queries. Yet, unless Statistics New Zealand has both checked to see that the combination of exact answers will not lead to any disclosure of confidential data, and further that the answers based on perturbed data are consistent with all exact answers, they may face additional threats. Therefore they could benefit from the model proposed in this work.

We show (see Section 4.1) that one cannot *independently* combine query restriction with perturbation, since that would lead to additional threats to security if the perturbation answers are not consistent with the exact answers. In a typical database setting there should be a history of important (often asked) queries. For instance it may be deemed important that queries asking for pairwise correlations between the confidential fields are answered exactly. The same may be true of such common query types such as MEAN, MEDIAN, and COUNT over all major subject categories (e.g. Doctors, Nurses, etc.). Then if there are f confidential fields and c major subject categories, a total of $O(f^2 + c)$ queries would be candidates for exact solution. For example, in the 2001 New Zealand census, around 400 queries for each of 76 districts and 19 geographical regions were considered. In each group of 400 queries, five queries are MEAN and MEDIAN and the rest are COUNT queries. Exact answers to MEAN and MEDIAN queries and perturbed answers to COUNT queries were published except for regions or districts with less than six individuals. All other queries would be considered in the run time stage. In our model an optimal subset of these are answered exactly as possible in the design stage. Then a perturbed database that is consistent with these exact answers is generated. In the run time stage ad hoc queries arrive and are answered based on the perturbed confidential data.

This overall scheme offers many benefits. Every query is answered and the user is notified when an answer is guaranteed to be exact. Linear as well as nonlinear queries, on both numerical and categorical data, can be handled. Since all query answers, whether exact or inexact, are consistent with the perturbed database the statistical properties can be guaranteed as in standard perturbation techniques. Finally there is no danger from collusion. The only significant downside, as compared to previous perturbation techniques, is that there is some extra computational burden in performing “consistent” perturbations. On the other hand, this burden will only occur at the design stage and is not particularly onerous (see Section 6). A subtle but very important benefit is that answering queries exactly in the design stage leads to better quality perturbed answers in the run stage.

1.1. Exact versus Stochastic Protection

A database can contain any number, say f , of confidential fields. If there are s subjects, so that each confidential vector is of size s , we can combine the f subvectors into a single vector by stacking the subvectors on top of each other. This vector, of size $n = sf$, is denoted by a and can contain any combination of numeric (e.g. salary) or categorical (e.g. credit status) data. The only caveat in treating the set of confidential fields as a singleton is that it may be important to preserve relationships, e.g. correlations, among the original fields in the perturbation phase. This concern can be handled easily (see Section 2.3 for details). For ease of exposition the primary presentation is for numeric data. Analysis for categorical data is given as appropriate throughout the paper.

Let $N := \{1, \dots, n\}$ denote the set of indexes of the confidential values (entries) in the stacked vector a . A general query maps the vector a to a real number. The queries we study are of the form $q(a, L)$, where $L \subset N$, and they entail a computation using only the entries of a indexed by the set

L . The *size* of a query is the cardinality of the support set L , i.e. the set required to compute the query.

Exact deterministic protection requires that the user cannot exactly determine any element of a from the answers to queries. To define stochastic protection, suppose that it is publicly known that the confidential datum a_i lies inside a bounded interval $[l, u]$. Let X_i be a random variable representing the user’s estimate of a_i within the interval (in Section 3 we indicate how the user may determine this probability). A natural definition is that a_i is stochastically protected if

$$\Pr \{X_i \in [r, s]\} \leq \delta, \forall [r, s] \subset [l, u], s - r \leq \epsilon, \quad (1)$$

where $\epsilon \geq 0$ and $\delta \in (0, 1]$ are two given constants, for all $i \in N$. That is, a_i must not be randomly estimated in any interval of range ϵ or smaller with probability δ or greater. It follows from (1) that stochastic protection implies deterministic protection for any a_i . Observe that the definition naturally extends to categorical data, for instance, for binary data, by using $[l, u] = [0, 1]$ and $\epsilon = 0$. It is also true, as is shown in Section 4, that inexact deterministic knowledge, i.e. the knowledge that a_i lies in a given interval, may lead to an unacceptable stochastic threat. We examine this type of threat in Section 4 and focus on exact deterministic protection in Sections 2 and 3. Finally, notice that the parameters δ and ϵ could depend on the confidential value a_i , that is, there is a pair (δ_i, ϵ_i) for each confidential value a_i such that (1) must be satisfied. However, we choose to use constant δ and ϵ to simplify the presentation.

1.2. An Overview

A general definition of stochastic protection of confidential data has been given in (1). A protection scheme that satisfies (1) for all confidential data will be deemed “secure.” Here we give an overview of the complete process to obtain security and, at the same time give the “best possible information.” The various process stages are described in each section of the paper. To avoid cumbersome notation, within every subsection the process (and resulting data set) will be referred to as “safe” if it satisfies the criteria of that subsection. The word “secure” is reserved for the ultimate level of protection (1).

Section 2 deals with the first phase of the design stage, namely determining the subset S of the candidate query set T to be answered exactly. Note that S may contain both linear and nonlinear queries. The initial focus is on linear queries and the set is deemed safe if the knowledge of the answers cannot yield any confidential datum exactly. The problem of finding a best safe set is shown to have a matroid intersection structure and a “greedy” approximate algorithm is given for the solution of instances deemed too large to be solved exactly. In Section 2.3 it is demonstrated that a similar model, and resulting algorithm, is obtained if S can contain nonlinear queries. The only caveat is that the set cannot be guaranteed safe if S contains nonlinear queries that can be combined in a nonlinear fashion to obtain confidential data. That is, safety is assured if only linear combinations of queries are employed by the user. In Section 3 it is observed that the exact answers of Section 2 can be used, via a uniform probability distribution, to calculate intervals that result in a possibly unacceptable stochastic threat. That threat can be obviated by removing one or more of the queries previously assigned to S . This is achieved by means of an algorithm that guarantees safety but is heuristic in that the problem of finding the optimal set to remove, which in itself is \mathcal{NP} -hard, is not solved optimally. At the conclusion of this step the resulting set S is secure, since even the remaining gap left after the steps of the previous section has been removed.

Once the set S has been determined, the question remains of how to answer the rest of the queries via perturbation of the confidential data. In Section 4 it is shown that the perturbation must be consistent with the answers to the queries in S or a threat results. Then two ways to determine consistent perturbations of the confidential vector are presented. It is also shown that a form of stochastic threat, similar to that described in Section 3, can also result from the perturbation. As is

done in Section 3, a method is presented to combat this type of threat. At the end of the process described in Section 4, the choice of queries for set S (potentially containing both linear and nonlinear queries) and the perturbed vector guarantee that all records are stochastically protected and that this level of protection cannot be circumvented by any inference scheme. Therefore the overall process is secure. In Section 5 attention is shifted to the run time stage in which all queries are answered via the perturbed vector. In addition there exists the safe possibility of providing exact answers to “altered” queries, i.e. queries that are “close” to those desired. The computational results of Section 6 indicate that the overall method can be extremely effective; answering many queries exactly and providing extremely good answers, in general, to the rest. The results also indicate that the proposed approach provides more accurate query answers than the standard perturbation approach, which follows intuitively since the perturbed vector is constrained to have additional characteristics in common with the original data set beyond simple addition of random noise.

2. Design Stage Phase I: Exact Answers

Here we provide mathematical models to solve the first subproblem of providing deterministic, exact protection. It is modeled as the problem of finding an optimal set of queries that can be answered exactly for a given set of weights assigned to the queries, and is shown to have a matroid intersection structure. Since even this subproblem is \mathcal{NP} -hard, we develop an efficient greedy heuristic and corresponding bounds on the objective function. We consider in separate subsections the cases of linear queries and the general case that includes nonlinear queries.

2.1. Linear Queries

We deal first with the case of linear queries because SUM and MEAN, for example, represent two of the most common query types. Also, the results for linear queries extend naturally to the case of general nonlinear queries.

2.1.1. Matroid Intersection Model The DBM chooses a set $S \subset T := \{q_1, \dots, q_m\}$ of linear target queries. Again to simplify the notation, let M be the index set $\{1, \dots, m\}$. Since each linear query is of the form $q(x) := q^T x$, where $q \in \mathbb{R}^n$, and completely defined by the vector q , we will not distinguish between a linear query and its n -dimensional defining vector. To illustrate the definition of a safe query set, suppose that we want to answer a collection of queries evaluated at a , but we do not want an intruder to learn any coordinate value of a by combining the queries. For example, if $n = 3$, $a = [2, 5, 8]^T$, and we answer the queries $q_1^T a := a_1 + a_2 + a_3 = 15$ and $q_2^T a := a_1 + a_2 = 7$, a user could learn the value of a_3 by solving the system

$$\begin{aligned} x_1 + x_2 + x_3 &= 15, \\ x_1 + x_2 &= 7. \end{aligned}$$

Solving such a system is equivalent to finding a linear combination of the query vectors $q_1 := [1, 1, 1]^T$ and $q_2 := [1, 1, 0]^T$ that yields the canonical vector $e_3 = [0, 0, 1]^T$. For instance, $q_1 - q_2 = e_3$. Therefore, a set of queries is safe if it is not possible to express any canonical vector as a linear combination of the queries in the set. A similar argument applies to linear queries applied to categorical data (e.g. count queries on binary data).

Notice that if the queries are linear, it is enough to consider disclosures arising from linear combinations because the users will at most know that the actual data belongs to an affine linear subspace. Therefore, any transformation (not necessarily linear) leading to a disclosure of a specific value of the database must project the data subspace into the canonical space corresponding to that coordinate. In other words, when restricted to the data subspace, any transformation leading to a disclosure must be a linear projection and must coincide with a linear combination of the queries yielding a canonical vector.

Formally, given a finite set of linear queries $T := \{q_j : j \in M\}$, a subset $K \subset M$, and an integer $i \in N$, we say that K is safe with respect to i if the canonical vector e_i cannot be expressed as a linear combination of the queries with indexes in K . We say that K is safe in general if K is safe with respect to all canonical vectors. Moreover, we denote by \mathcal{F} the collection of all safe sets derived from M , that is,

$$\mathcal{F} := \{K \subset M : K \text{ is a safe set}\}.$$

The DBM assigns positive weights to each query q_j , $j \in M$, and then maximizes the total weight of the queries she decides to answer exactly. For example, such weights can be assigned based on user demand for queries or, if the DBM does not have a mechanism to ascertain weights on queries, she can assign unit weights to all queries in T .

In summary, our goal can be reformulated as finding a subset $K^* \subset M$ solving the following combinatorial optimization problem

$$\text{(OPT)} \quad z_1 = \max \left\{ \sum_{j \in K} w_j, K \in \mathcal{F} \right\}.$$

For convenience, we sometimes use the notation

$$W(K) := \sum_{j \in K} w_j,$$

for all subsets K of M .

To illustrate, consider a database consisting of $n = 4$ confidential values and T consisting of the four SUM queries given below with $(w_1, \dots, w_4) = (40, 20, 20, 30)$.

$$T := \{q_1, q_2, q_3, q_4\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}. \quad (2)$$

Then, for example, $\{q_2, q_4\}$ is unsafe with respect to $i = 4$ since $q_4 - q_2 = e_4$, although it is safe with respect to $i = 1, 2, 3$. In this case, the optimal safe set is given by $K^* = \{1, 4\}$ with optimal objective value $W(K^*) = 70$.

Chin and Ozsoyoglu (1982) study a related problem to OPT. Given a finite set of linear queries T , their problem consists of finding a maximum cardinality subset of T that can be answered exactly without compromising the database in the linear sense described above. They show that a corresponding Yes-No decision version of the problem (Is there an answerable subset of queries with cardinality greater than or equal to a given positive integer k ?) is \mathcal{NP} -complete, and so, their original problem is \mathcal{NP} -hard. Clearly, if we let $w_j = 1$ for all $j \in M$, Chin and Ozsoyoglu's problem is polynomially reduced to OPT, so that OPT is \mathcal{NP} -hard. Hence, finding exact solution sets to OPT could be computationally expensive. On the other hand, we provide an efficient algorithm to find extremely good approximate solutions in Section 2.2.

Before considering solutions to OPT, we first state a theorem showing that it is equivalent to a weighted matroid intersection problem. The proof of the theorem can be found in the Appendix.

THEOREM 1. *For a fixed integer $i \in N$, consider the family \mathcal{F}_i of subsets of M defined as*

$$\mathcal{F}_i := \{K \subset M : K \text{ is a safe set w.r.t. } i\}.$$

Then, (M, \mathcal{F}_i) is a matroid for each $i \in N$. Furthermore, by noticing that

$$\mathcal{F} = \bigcap_{i=1}^n \mathcal{F}_i,$$

it follows that \mathcal{F} is an intersection of n matroids.

Table 1 Independent (Safe) Sets.

K	\mathcal{F}_1	\mathcal{F}_2	\mathcal{F}_3	\mathcal{F}_4	$W(K)$
\emptyset	x	x	x	x	0
{1}	x	x	x	x	40
{2}	x	x	x	x	20
{3}	x	x	x	x	20
{4}	x	x	x	x	30
{1,2}	x	x	x	x	60
{1,3}	x		x	x	60
{1,4}	x	x	x	x	70
{2,3}	x	x	x	x	40
{2,4}	x	x	x		50
{3,4}	x	x	x	x	50
{1,2,3}	x			x	80
{1,2,4}		x	x		90
{1,3,4}	x		x	x	90
{2,3,4}	x	x	x		70
{1,2,3,4}					110

For example, using the data (2), we obtain Table 1 that shows which subsets of M are independent (safe) sets with respect to each of the matroids (M, \mathcal{F}_i) , and the total weight for each subset. It is clear that the optimal safe set is $K^* = \{1, 4\}$.

2.2. Approximate Solutions

If m or n are considered too large to find an exact solution to OPT, we can exploit the matroid intersection structure of our model to devise an approximation algorithm. That is, the algorithm will yield a safe set that may not have optimal weight. Let $r_i(K)$ be the rank of the subset K in the matroid (M, \mathcal{F}_i) , $i \in N$. The rank function is computed by using the following greedy algorithm (see Nemhauser and Wolsey 1988).

ALGORITHM 1. Given K and i , the algorithm computes $r_i(K)$.

Initialization step: Set the index set $R = \emptyset$ and $r = 0$.

Iteration step:

1. Pick an arbitrary $k \in K$ and remove k from K .
2. If query q_k is not a linear combination of $\{q_j : j \in R\}$ and e_i is not a linear combination of $\{q_j : j \in R \cup \{k\}\}$, then add k to the set R and increase r by one.

Stopping rule: Stop when K becomes empty; the value of $r_i(K)$ is r .

Notice that the iteration step is executed $|K|$ times, and in each iteration the most computationally expensive step is testing the conditions from the if-statement in step 2. To test the conditions, it is necessary to solve two linear systems, respectively, each consisting of n equations and at most $|K|$ unknowns. Hence, the values $r_i(K)$ can be efficiently computed in polynomial time by using Algorithm 1.

We also let

$$r(K) := \min_{i \in N} r_i(K),$$

for all $K \subset M$. For example, for the data (2), Table 2 shows the rank of the subsets of M within each of the four matroids (M, \mathcal{F}_i) as well as the value of the function r evaluated at each subset. Clearly, when $r(K) = |K|$, it follows that K is an independent (safe) set for all the matroids, that is, K is an independent set in the matroid intersection.

Following Nemhauser and Wolsey (1988), let

$$r_i^D(K) := |K| + r_i(M \setminus K) - r_i(M),$$

Table 2 Rank Evaluation.

K	r_1	r_2	r_3	r_4	r
\emptyset	0	0	0	0	0
$\{1\}$	1	1	1	1	1
$\{2\}$	1	1	1	1	1
$\{3\}$	1	1	1	1	1
$\{4\}$	1	1	1	1	1
$\{1,2\}$	2	2	2	2	2
$\{1,3\}$	2	1	2	2	1
$\{1,4\}$	2	2	2	2	2
$\{2,3\}$	2	2	2	2	2
$\{2,4\}$	2	2	2	1	1
$\{3,4\}$	2	2	2	2	2
$\{1,2,3\}$	3	2	2	3	2
$\{1,2,4\}$	2	3	3	2	2
$\{1,3,4\}$	3	2	3	3	2
$\{2,3,4\}$	3	3	3	2	2
$\{1,2,3,4\}$	3	3	3	3	3

for all $i \in N$ and $K \subset M$. The function r_i^D is the rank function of the dual matroid associated with (M, \mathcal{F}_i) . Hence, if we let

$$f(K) := \sum_{i=1}^n r_i^D(K),$$

it follows from Nemhauser and Wolsey (1988), page 709, that f is a submodular function and

$$z_1 = \sum_{j \in M} w_j - \min \left\{ \sum_{j \in K} w_j : f(K) = f(M), K \subset M \right\}. \quad (3)$$

Therefore, by using a greedy heuristic we can obtain an approximate solution, denoted by \hat{K} , to the minimization problem in (3), and use the bounds discussed in Nemhauser and Wolsey (1988), page 712, to evaluate the resulting approximation. The algorithm is as follows (we use the convention $w/0 = \infty$, for $w \neq 0$).

ALGORITHM 2. Matroid Intersection Greedy (MIG) algorithm to compute an approximate solution set \hat{K} .

Initialization step: Set the index sets $K^0 = \emptyset$, $M^1 = M$, and $t = 1$.

Iteration step:

1. Find index j_t where $\min\{w_j / (f(K^{t-1} \cup \{j\}) - f(K^{t-1})) : j \in M^t\}$ is attained.
2. Set $M^{t+1} = M^t \setminus \{j_t\}$, $K^t = K^{t-1} \cup \{j_t\}$, and increase t by one.

Stopping rule: Stop when $f(K^t) = f(M)$; an approximate solution \hat{K} is K^t .

The algorithm starts with the full set M and an empty set K . In every iteration it removes one index of M corresponding to an unsafe query until it reaches a safe set. Every removed query is added to the set K . According to the definition of the submodular function f , a subset $M \setminus K$ is safe if and only if $f(K) = f(M)$. Hence, our stopping criterion is that the set of removed queries K satisfies $f(K) = f(M)$. Observe that as the set K increases by removing queries from M and adding them to K , $f(K)$ also increases until it reaches $f(M)$ and the algorithm stops. The queries can be removed in any order, but in algorithm MIG we use a greedy approach, that is, the algorithm removes one query of M that minimizes the ratio

$$\frac{w_j}{f(K^{t-1} \cup \{j\}) - f(K^{t-1})}.$$

Table 3 Dual Rank Evaluation.

K	r_1^D	r_2^D	r_3^D	r_4^D	f
\emptyset	0	0	0	0	0
{1}	1	1	1	0	3
{2}	1	0	1	1	3
{3}	0	1	1	0	2
{4}	1	0	0	1	2
{1,2}	1	1	1	1	4
{1,3}	1	1	1	0	3
{1,4}	1	1	1	1	4
{2,3}	1	1	1	1	4
{2,4}	1	0	1	1	3
{3,4}	1	1	1	1	4
{1,2,3}	1	1	1	1	4
{1,2,4}	1	1	1	1	4
{1,3,4}	1	1	1	1	4
{2,3,4}	1	1	1	1	4
{1,2,3,4}	1	1	1	1	4

Table 4 Results from the Approximation Algorithm.

t	M^t	j_t	K^t	$f(K^t)$
1	{1, 2, 3, 4}	2	{2}	3
2	{1, 3, 4}	3	{2, 3}	4

Notice that the ratio decreases as the weights w_j decrease and as the difference $f(K^{t-1} \cup \{j\}) - f(K^{t-1})$ increases. Roughly speaking, the algorithm removes queries that do not contribute too much to the total weight and provide a large improvement in the evaluation of the function f (thus hopefully getting faster to the goal of reaching $f(M)$).

From Theorem 9.4 in Nemhauser and Wolsey (1988) (see also Nemhauser and Wolsey 1978, Nemhauser et al. 1978), we obtain the following result.

THEOREM 2. *Let \hat{K} be the set returned by the MIG algorithm and $z_G := \sum_{j \in \hat{K}} w_j = W(\hat{K})$. Then, $M \setminus \hat{K}$ is a safe set and*

$$\frac{z_G}{W(M) - z_1} \leq H(d) := \sum_{i=1}^d \frac{1}{i} \leq 1 + \ln d, \quad (4)$$

where $d = \max\{f(\{j\}) : j \in M\}$.

Since $M \setminus \hat{K}$ from Theorem 2 is a safe set, we clearly obtain a lower bound for the optimal value z_1 of OPT, namely,

$$W(M \setminus \hat{K}) \leq z_1.$$

Furthermore, the harmonic number inequality (4) yields an upper bound for z_1 , namely,

$$z_1 \leq W(M) - \frac{z_G}{H(d)}.$$

In other words, we obtain the bounds

$$W(M) - W(\hat{K}) \leq z_1 \leq W(M) - \frac{W(\hat{K})}{H(d)}. \quad (5)$$

For example, in Table 3 we show the computation of the dual rank functions r_i^D as well as the value of the function f for every subset of M . It follows that d from Theorem 2 is 3, and so, the harmonic number for $d = 3$ is $H(3) = \sum_{i=1}^3 1/i = 11/6 \approx 1.83$ ($1 + \ln d \approx 2.1$). Table 4 shows the

result of applying the MIG algorithm to our data. Since $f(M) = 4$, the algorithm stops after two iterations. The set found by the algorithm is $\hat{K} = \{2, 3\}$ with a weight of $W(\hat{K}) = 40$. In this case, since $M \setminus \hat{K} = \{1, 4\}$, the solution found by the algorithm is optimal. If we did not know the optimal objective value z_1 , using the bounds (5), and since $W(M) = 110$, we would find

$$110 - 40 = 70 \leq z_1 \leq W(M) - (W(\hat{K})/H(3)) \approx 88.1,$$

so that $z_1 \in [70, 88.1]$.

For a small example for which the MIG algorithm does not yield an optimal solution, consider the following,

$$T := \{q_1, q_2, q_3, q_4\} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 8 \\ 8 \\ 2 \end{bmatrix} \right\}; \quad (6)$$

with weights $w_1 = 19$, $w_2 = 10$, $w_3 = 10$, and $w_4 = 20$, respectively. It is not difficult to see that the optimal solution to the corresponding OPT problem is $K^* = \{2, 3, 4\}$ with objective value $W(K^*) = z_1 = 40$. The solution provided by the algorithm is $\hat{K} = \{2, 3\}$, and so, the approximate solution derived from the algorithm is $M \setminus \hat{K} = \{1, 4\}$ with objective value $W(\hat{K}) = 39$. Since for the data (6) we have $W(M) = 59$, $d = \max\{f(\{j\}) : j \in M\} = 4$, and $H(4) = 25/12 \approx 2.1$, the corresponding bounds (5) are given by

$$59 - 39 = 20 \leq z_1 \leq W(M) - (W(\hat{K})/H(3)) \approx 40.4,$$

so that $z_1 \in [20, 40.4]$ in this example.

Finally, by using the *LUP* matrix decomposition for solving linear systems of equations (see Cormen et al. 2000, Golub and Van Loan 1997), the MIG algorithm can be efficiently implemented. In Proposition 1, we present a polynomial upper bound on the number of arithmetical operations required by the algorithm. The proof of the proposition can be found in the Appendix.

PROPOSITION 1. *Algorithm MIG can be implemented in no more than $O(m^4 n^2)$ arithmetical operations.*

2.3. Linear Combinations of Nonlinear Queries

Here a set is made safe even if it contains nonlinear queries. However we continue to limit the analysis to linear combinations of such queries. Of course nonlinear combinations of nonlinear queries are possible and could be dangerous. A simple example is when exact answers are provided to the queries $q_1(a_1, a_2) := a_1^2 - a_2^2$ and $q_2(a_1, a_2) := a_1 + a_2$. The user can then divide the first answer by the second to get $a_1 - a_2$ and thus determine both confidential data. The result of the analysis of Section 3 will be a device that protects against all linear and nonlinear combinations of queries of any type!

As before we start with a finite target set $T := \{q_j : j \in M\}$ of any mix of linear and nonlinear queries, where M is a nonempty index set. Examples of nonlinear queries include such standard ones as VARIANCE and REGRESSION. Another example pertains to the issue raised in Section 1.1 of stacking multiple confidential fields in the presence of correlation among them. If it is required to keep that correlation intact, it can be one of the nonlinear queries discussed here. For example, if the database is of the form $[u, v]$ where u and v are s -dimensional numerical subvectors, then computing a correlation query between u and v is equivalent to applying to the stacked vector

$$a := \begin{bmatrix} u \\ v \end{bmatrix},$$

the function

$$q(a) := \frac{\sum_{i=1}^s \left(a_i - \frac{1}{s} \sum_{j=1}^s a_j \right) \left(a_{i+s} - \frac{1}{s} \sum_{j=s+1}^{2s} a_j \right)}{\left(\sum_{i=1}^s \left(a_i - \frac{1}{s} \sum_{j=1}^s a_j \right)^2 \right)^{1/2} \left(\sum_{i=s+1}^{2s} \left(a_i - \frac{1}{s} \sum_{j=s+1}^{2s} a_j \right)^2 \right)^{1/2}}.$$

The queries in T map the n -dimensional real space into \mathfrak{R} . As with linear queries we define a threat as the possibility for an intruder to learn the confidential value by using a linear combination of the answers provided to a set of queries. A threat exists if the intruder could linearly combine the answers to a set of nonlinear queries to deduce an individual confidential value.

Let $\pi_i(x)$ be the n -dimensional canonical projection mapping, that is, $\pi_i(x) := x_i$ for all $x \in \mathfrak{R}^n$. A subset $K \subset M$ is safe with respect to i if π_i cannot be expressed as a linear combination of the queries with indexes in K . We denote by $\langle q_j : j \in K \rangle$ the set of linear combinations of the queries with indexes in K . Hence, K is safe with respect to i if $\pi_i \notin \langle q_j : j \in K \rangle$.

We define \mathcal{F} and \mathcal{F}_i as in Section 2.1. Notice that the proof of Theorem 1 still stands if, instead of using linear queries and canonical vectors, we use nonlinear queries and canonical projections. Therefore, we still have that (M, \mathcal{F}_i) is a matroid for each i and that \mathcal{F} is an intersection of n matroids.

The rank function is computed by using the following algorithm, which is a reformulation of Algorithm 1 for arbitrary queries.

ALGORITHM 3. Given K and i the algorithm computes $r_i(K)$.

Initialization step: Set the index set $R = \emptyset$ and $r = 0$.

Iteration step:

1. Pick an arbitrary $k \in K$ and remove k from K .
2. If $q_k \notin \langle q_j : j \in R \rangle$ and $\pi_i \notin \langle q_j : j \in R \cup \{k\} \rangle$, then add k to the set R and increase r by one.

Stopping rule: Stop when K becomes empty; the value of $r_i(K)$ is r .

The algorithm is easily implementable except for the step in which the condition

$$q_k \notin \langle q_j : j \in R \rangle \text{ and } \pi_i \notin \langle q_j : j \in R \cup \{k\} \rangle, \tag{7}$$

is tested. In the case of linear queries the implementation of this step reduces to finding the solution to two systems of linear equations, which can be efficiently done by using a standard *LUP* matrix decomposition method (see proof of Proposition 1). However, in the context of nonlinear queries implementing this step is more complex.

To implement condition (7) we follow a least-squares approach (see Luenberger 1968) in which we provide a random sample Y consisting of s n -dimensional linearly independent points to evaluate the queries in T , and where s satisfies $\min\{n, t\} \leq s \leq n$. Then, testing the condition reduces to solving a linear system of equations in terms of the matrix of query evaluations at each sample point in Y . The technical details are given in the proof of Proposition 2 in the Appendix, along with an illustrative example.

PROPOSITION 2. *Algorithm MIG can be implemented in no more than $O(m^4 n^2 s^2)$ arithmetical operations, where s is the size of the sample Y used in evaluating the condition (7) for the queries in the set T .*

3. Stochastic Threat From Deterministic Intervals

In this section we extend the previous analysis in two ways. The obvious extension is to consider stochastic interval protection instead of restricting ourselves to deterministic exact protection. It follows, however, that if the former is guaranteed, for any intervals surrounding the confidential data

and for any parameters of stochasticity then the latter is likewise guaranteed. Thus the result of adding the technique of this section is that any gap, in terms of protection, from the previous two sections will be removed. The only such gap was the possibility of nonlinear combinations of general queries as discussed in the previous section.

Assume that a solution, either optimal or suboptimal to OPT, has been determined. Then, if a bound on the confidential data is generally known, the user can calculate deterministic bounds on the a vector as discussed in Gopal et al. (1998). Furthermore these bounds can be used to update the user's probability function over that vector and thereby possibly violate (1). For instance, if it is known that $a \geq 0$, then the user can solve the following mathematical program for any $i \in N$,

$$\begin{aligned} \text{(G3)} \quad & \max \quad z_i(K) = u_i - l_i, \\ & \text{s.t.} \\ & \quad Q_K(u) = b, \\ & \quad Q_K(l) = b, \\ & \quad l, u \geq 0; \end{aligned}$$

where $b := Q_K(a)$ and $Q_K(x)$ is the $|K|$ dimensional vector whose entries are the values $q_j(x)$ for all $j \in K$ and $x \in \mathfrak{R}^n$. If $z_i^*(K) := u_i^*(K) - l_i^*(K)$ is optimal to G3, the user will know that the confidential value a_i is in the interval $[l_i^*(K), u_i^*(K)]$. Notice that in the special case of linear queries, G3 is a linear program.

Clearly, if $z_i^*(K) \leq \epsilon$, then (1) is violated. On the other hand, even if $z_i^*(K) > \epsilon$ it is possible to argue that the bounds $l_i^*(K), u_i^*(K)$ may cause violation of (1). The user is able to determine that $a_i \in [l_i^*(K), u_i^*(K)]$, but has no additional information to make one point in the interval more likely than another. Therefore it is reasonable to assume that, in the absence of any other information, the user's probability density for her random estimate X_i of a_i is uniform over $[l_i^*(K), u_i^*(K)]$. Then it follows that if

$$u_i^*(K) - l_i^*(K) \geq \epsilon/\delta, \tag{8}$$

for all $i \in N$, then the user's probability that X_i is contained in a given interval of range no greater than ϵ is less than δ , and so the stochastic protection condition (1) is satisfied for all $i \in N$. Notice that if condition (8) is satisfied, the worst case scenario of what a user can infer from the released exact answers is an n -dimensional box with volume $(\epsilon/\delta)^n$ containing the confidential vector, regardless of what type of linear or nonlinear queries are exactly answered and regardless of how they are combined. Therefore, by solving problem G3 and enforcing condition (8), our method provides an extra safety against inference threats.

In summary, the previous discussion suggests the following optimization problem in order to satisfy the stochastic protection condition:

$$\text{(OPT2)} \quad \max\{W(K) : K \subset N \text{ and } u_i^*(K) - l_i^*(K) \geq \epsilon/\delta \text{ for all } i \in N\},$$

where $l_i^*(K)$ and $u_i^*(K)$ are optimal solutions to the problem G3 corresponding to the set K .

By setting $\epsilon = 0$ and $\delta = 1$, it is clear that problem OPT polynomially reduces to an instance of problem OPT2. Therefore, problem OPT2 is \mathcal{NP} -hard. An option for the DBM in this case is to use the following greedy heuristic to find an approximate solution to OPT2. That is, given a safe set K , systematically remove a query in K with the least weight until (8) holds for all $i \in N$. Such a heuristic could be executed using an optimal solution or an approximate solution to OPT from algorithm MIG. The complexity of the heuristic will depend on how difficult is to solve the associated G3 subproblems. In case of linear queries and/or MAX-MIN queries, each G3 subproblem reduces to a collection of linear problems that can be polynomially solved (Gopal et al. 1998). For example,

for the queries $q_1(a_1, a_2) := a_1 + a_2$ and $q_2(a_1, a_2) := \min\{a_1, a_2\}$, the corresponding G3 problem can be solved by combining the solutions to the linear programs

$$\begin{array}{ll} \min z = x_i, & \max z = x_i, \\ x_1 + x_2 = b_1, & x_1 + x_2 = b_1, \\ x_j = b_2, & x_j = b_2 \\ x_1, x_2 \geq b_2, & x_1, x_2 \geq b_2, \end{array} \quad (9)$$

for $i = 1, 2$ and $j = 1, 2$, where $b_1 := a_1 + a_2$ and $b_2 := \min\{a_1, a_2\}$.

As for the quality of the solution provided by the greedy heuristic from the previous paragraph, let K_{OPT} be an optimal solution to OPT, K_{OPT2} be an optimal solution to OPT2, K_{MIG} be the solution returned by algorithm MIG, and K_{QR} be the set obtained from $M \setminus K_{\text{MIG}}$ by using the query-removal heuristic. Then, clearly from (5) we have

$$W(K_{\text{QR}}) \leq W(K_{\text{OPT2}}) \leq W(K_{\text{OPT}}) \leq W(M) - \frac{W(K_{\text{MIG}})}{H(d)},$$

where $H(d)$ is as in (4).

4. Design Stage Phase II: Consistent Perturbation

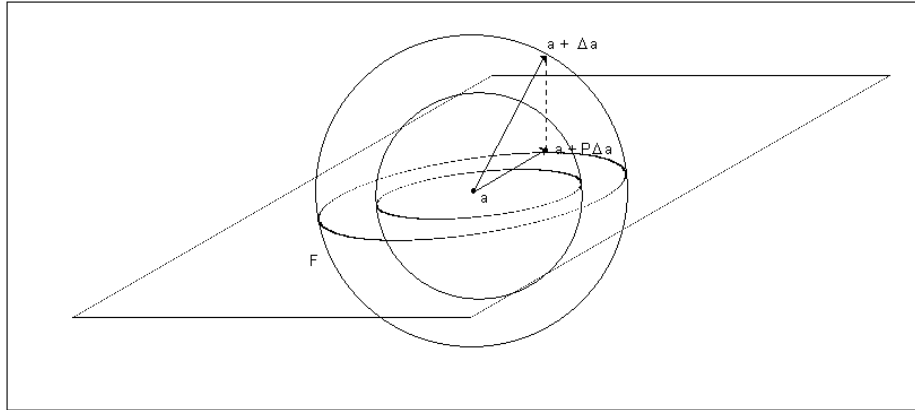
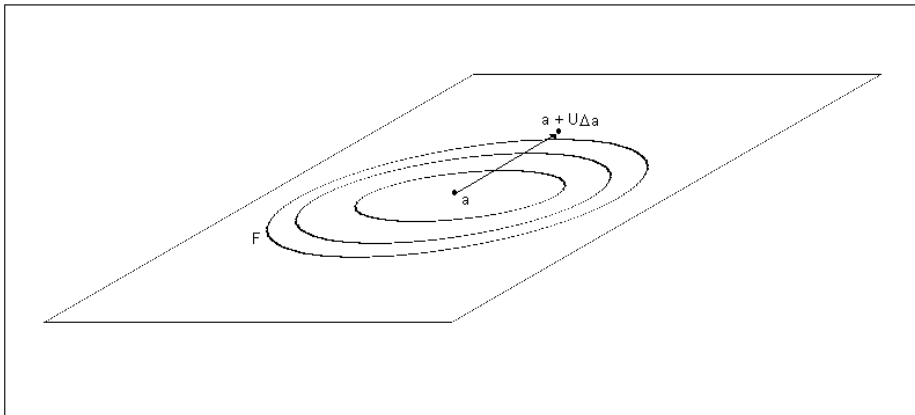
In this section we develop two random data perturbation methods that provide answers consistent with the exact answers to the queries in a given set S , and then we discuss how to provide stochastic protection with our methods. If S is empty, then our methods become standard perturbation methods such as those discussed in Traub et al. (1984). The perturbed vector that is created guarantees that all records are stochastically protected and thus that the entire process is secure. As mentioned earlier the perturbed vector can be released to the users or used as the basis from which to answer queries. In addition the user will be told that answers to queries in S are exact.

4.1. The Need for Consistency

Consistency is a property associated with the algorithm that gives answers to queries. That is, an algorithm is consistent if a given query is answered the same way every time it is asked. In our model consistency is of paramount importance because the queries in S are guaranteed to be answered exactly. Suppose a user is intent upon determining the confidential datum a_i . Then the user can take any query $q \in S$ and ask the new query $q' = q + e_i$. If $q' \notin S$ and the exact answers and perturbed answers to queries are not consistent, the answer to q' along with the exact answer to q yields an estimate of a_i . If this process is repeated k times with different queries q , the user then has access to $k + 1$ linearly independent estimates of a_i where the last estimate comes from asking the singleton query e_i . When k is large, an extension of the weak law of large numbers (see Lehmann 1998, Chapter 2) shows that the arithmetic mean of the estimates will be close to the actual confidential value with high probability. On the other hand if consistency is preserved the user can obtain only one estimate of the confidential value.

4.2. Consistent Perturbations for Linear Queries

As usual, we first consider the case of linear queries. Without loss of generality we assume that S consists of a non-empty set of linearly independent queries. Formally, the DBM will determine a random vector perturbation Δa and either provide the vector $a + \Delta a$ to the user or give the answer $q(a + \Delta a)$ to any other query q not in S . To ensure consistency we impose the constraint $q^T(a + \Delta a) = q^T a$, or equivalently, $q^T \Delta a = 0$, for every query q in S . Hence, the answers to the queries in S computed by the user from the perturbed data will be the same as the answers released via the techniques of Sections 2 and 5.1 by the DBM.

Figure 1 Perturb and Project onto Hyperplane.**Figure 2** Perturb in the Hyperplane Direction.

Again we denote by K the index set corresponding to the queries in S , that is $q_j \in S$ if and only if $j \in K$, by $Q := Q_K$ the $n \times |K|$ matrix whose columns are the queries with index in K , and by \mathcal{H}_S the affine hyperplane generated by those queries, that is

$$\mathcal{H}_S := \{x \in \mathfrak{R}^n : q_i^T x = q_i^T a, \forall i \in K\}.$$

Notice that because of the linear independence assumption the hyperplane \mathcal{H}_S has dimension $n - |K|$.

We propose two perturbation methods that will preserve the consistency in the answers as discussed above:

Method 1: Projection onto hyperplane. Randomly perturb the data using an n -dimensional elliptically symmetric probability distribution (e.g. a multivariate normal distribution), and then project the perturbed data onto the affine hyperplane \mathcal{H}_S generated by the queries in S (see Figure 1); and

Method 2: Perturbation on the hyperplane direction. Randomly perturb the data in the direction of the affine hyperplane \mathcal{H}_S using an $(n - |K|)$ -dimensional elliptically symmetric probability distribution (see Figure 2).

To implement Method 1, let $r := Q^T a$ be the vector of exact answers to the queries with respect to the confidential data vector a . Let $F(x)$ denote an n -dimensional elliptically symmetric probability distribution and let Δa be a realization of a random vector variable with distribution F . Then Method 1 can be implemented by solving the well-known least-squares program

$$(LS) \quad \min \{ \|a + \Delta a - x\|_2 : Q^T x = r \},$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm. It follows that the projection is given by

$$\begin{aligned} x^* &:= a + P\Delta a, \\ P &:= I - QQ^+, \end{aligned}$$

where as before, $^+$ denotes the standard pseudo-inverse operator of a matrix (notice that P is the projection matrix onto the hyperplane \mathcal{H}_S). As such, we can answer queries by using the projected perturbation $a + P\Delta a$.

For example, suppose that $S = \{q_1, q_4\}$. Thus,

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix},$$

and so the projection matrix P is given by

$$P = \frac{1}{5} \begin{bmatrix} 2 & 1 & 1 & -2 \\ 1 & 3 & -2 & -1 \\ 1 & -2 & 3 & -1 \\ -2 & -1 & -1 & 2 \end{bmatrix}.$$

If $a = (2, 3, 3, 8)^T$ and the random realization is $\Delta a = (10, 10, 10, 10)^T$, then the perturbed vector is

$$a + P\Delta a = \begin{bmatrix} 6 \\ 5 \\ 5 \\ 4 \end{bmatrix}.$$

Clearly, $q_1^T(a + P\Delta a) = q_1^T a = 10$ and $q_4^T(a + P\Delta a) = q_4^T a = 14$.

Given that finding the least-squares solution to LS can be done very efficiently (see Golub and Van Loan 1997), Method 1 is attractive because of its computational simplicity. On the other hand, because the initial projection may be close to a , the guarantee of stochastic protection described in Section 4.4 may require more than one iteration.

Method 2 can be implemented as follows. Suppose that the matrix Q can be expressed as

$$Q = \begin{bmatrix} B \\ R \end{bmatrix},$$

where B is a $|K| \times |K|$ non-singular matrix and R is an $(n - |K|) \times |K|$ -dimensional matrix. Such an expression is possible, perhaps after permutation of a few rows, since Q has full-column rank. Any vector $x \neq 0$ such that $Q^T x = 0$ represents a non-trivial directional vector of the hyperplane. As such, notice that if we partition $x = [x_B, x_R]^T$, then x must satisfy

$$x_B = -B^{-T} R^T x_R.$$

Therefore, let U be the matrix defined as

$$U := \begin{bmatrix} -B^{-T}R^T \\ I_{n-|K|} \end{bmatrix}. \quad (10)$$

Hence, we first compute a realization Δa of an $(n-|K|)$ -dimensional non-singular multivariate normal distribution with zero mean and covariance matrix $\hat{\Sigma}$, and then we return the perturbation $a + U\Delta a$.

For example, using the same matrix Q as before, we have

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},$$

and so,

$$U = \begin{bmatrix} 0 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Hence, if as before $a = (2, 3, 3, 8)^T$ and the random realization is $\Delta a = (1, 1)^T$, then the perturbed vector is

$$a + U\Delta a = \begin{bmatrix} 1 \\ 1 \\ 4 \\ 9 \end{bmatrix}.$$

We again have $q_1^T(a + U\Delta a) = q_1^T a = 10$ and $q_4^T(a + U\Delta a) = q_4^T a = 14$.

Notice that in Method 2 we use a non-singular multivariate distribution. In this case, the distribution of the perturbation term $U\Delta a$ is also a multivariate normal with zero mean and covariance matrix $U\hat{\Sigma}U^T$. Also, because we are assuming that Q has full-column rank, the diagonal elements of $U\hat{\Sigma}U^T$ are always positive, and so we can compute a realization Δa so that the perturbation $a + U\Delta a$ is far from the confidential vector a , while still remaining on the hyperplane \mathcal{H}_S . The disadvantage of Method 2 is that finding the matrix U can be computationally more difficult than finding the projection matrix P from Method 1.

4.3. Nonlinear Queries and Categorical Data

In case the set S includes nonlinear queries, a similar method to Method 1 above can be used to determine a consistent perturbation with the exact answer queries. Let $r := Q(a)$ be the vector of exact answers to the queries with respect to the confidential data vector a ($r_i = q_i(a)$ for each query q_i in S). As before, we assume that the DBM uses an n -dimensional elliptically symmetric probability distribution F to compute a realization Δa of a random vector variable with distribution F . Then, the DBM solves the problem

$$(NLS) \quad \min \{ \|a + \Delta a - x\|_2 : Q(x) = r \}.$$

The complexity of solving problem NLS obviously depends on the queries in S . However, notice that for typical queries (e.g. SUM, AVERAGE, second moment queries such as VARIANCE, COVARIANCE, and CORRELATION, and order queries such as MAX and MIN) NLS can be solved as a collection of quadratic programs with linear and quadratic constraints (see Gopal et al. 1998, and example in (9)), which can be efficiently solved (see Bazaraa et al. 1993).

If the confidential vector includes binary data, then this method can be further extended. Let $L \subset N$ be the set of indexes corresponding to confidential binary entries in a and $\bar{L} := N \setminus L$ be the set of indexes corresponding to confidential numerical entries in a . First we compute a perturbation

Δa_L for the binary part of a by using standard techniques as in Abul-Ela et al. (1967), Warner (1965). We also compute a perturbation $\Delta a_{\bar{L}}$ for the numerical part of a using distribution F . Then we solve

$$(NBLs) \quad \min \{ \|a_L + \Delta a_L - x_L\|_2 + \|a_{\bar{L}} + \Delta a_{\bar{L}} - x_{\bar{L}}\|_2 : Q(x) = r, x_L \text{ binary} \}.$$

This analysis can similarly be extended to more general categorical data.

4.4. Stochastic Interval Protection

As indicated in Section 3 one type of stochastic threat arises from simply answering queries in S exactly. It is countered by imposing the condition (8). Here we provide other conditions from threats arising directly from the perturbations. For instance, when a linear query (not necessarily in S) is answered by using a randomly perturbed vector $a + \Delta a$, a user can use Chebyshev's inequality to determine a confidence interval for the true value of the query as long as the user knows the covariance matrix associated with the probability distribution used to generate Δa (the DBM has the option of releasing the covariance matrix to the users, in what follows we assume she does). For example, if Δa is generated using a multivariate normal distribution with zero mean vector and covariance matrix Σ , then Chebyshev's inequality reduces to

$$\Pr \{ |q^T \Delta a| \geq c \} \leq \frac{\text{Var}(q^T \Delta a)}{c^2} = \frac{1}{c^2} q^T \Sigma q, \quad (11)$$

for all $c > 0$. Therefore, the smaller the term $q^T \Sigma q$, the more certain the users are that the perturbed answer $q^T(a + \Delta a)$ is a good approximation to the actual answer $q^T a$. In particular, it is possible to obtain a confidence interval for any confidential value of the vector a since inequality (11) applied to any canonical (singleton) query yields

$$\Pr \{ |\Delta a_i| \geq \epsilon/2 \} \leq \frac{4\text{Var}(\Delta a_i)}{\epsilon^2} = \frac{4}{\epsilon^2} \Sigma_{ii}. \quad (12)$$

Therefore, the level of individual protection provided by the perturbation depends on the size of the diagonal elements of the covariance matrix Σ . Accordingly, the DBM should make Σ_{ii} large enough to avoid making the probability on the left of (12) too small.

On the other hand, making Σ_{ii} large does not necessarily ensure that the realization Δa_i satisfies $|\Delta a_i| > \epsilon/2$. In other words, there could still be a possibility of an interval disclosure threat. To avoid this, the DBM can compute a sequence of independent identically distributed perturbations (using either Method 1 or 2) until obtaining one realization that satisfies $|\Delta a_i| > \epsilon$ for all $i \in N$. If we let p_a be the following probability

$$p_a := \Pr \{ |\Delta a_1| > \epsilon/2, \dots, |\Delta a_n| > \epsilon/2 \},$$

and X be the random variable corresponding to the number of perturbations before obtaining one outside the multidimensional box $[a_1 - \epsilon, a_1 + \epsilon] \times \dots \times [a_n - \epsilon, a_n + \epsilon]$, then X has a geometric probability distribution with parameter p_a . Therefore, $E[X] = (1 - p_a)/p_a$, so that, on average the number of perturbations that need to be computed is $(1 - p_a)/p_a$.

In case the DBM uses a non-singular multivariate normal distribution with zero mean and covariance matrix Σ , probability p_a is easily computable by standard numerical integration methods. Moreover, it is common to use a distribution with Σ a diagonal matrix proportional to the identity matrix, say $\Sigma = \sigma^2 I$. Hence, probability p_a becomes

$$p_a = 2^n (1 - \varphi(\epsilon/(2\sigma)))^n,$$

where $\varphi(x)$ denotes the standard normal distribution. If the DBM wishes to use another probability distribution, then p_a can be estimated by using simulation.

5. Run Time Stage

At run time, ad hoc query q arrives and is answered with $q(a + \Delta a)$ based on the perturbed data. If $q \in S$ the user is told that the answer is guaranteed to be exact or can check for himself if the answers to queries in S are published.

5.1. Additional Exact Answers: An Additional Intermediate Stage

An option that may require some additional computation is to also offer an exact answer to a query “close” to q along with the answer based on the perturbed data. For example a user might ask for the “average salary of professors of German Literature” at a given university. If the query is not in S , the option of receiving an exact answer to the safe “average salary of professors of German or French Literature” query might be more appealing to the user than just receiving a perturbed answer to the original query.

To do that, the DBM may choose from a finite set $C(q)$ of queries reasonably close to q . Many measures of the degree of alteration of a query could be considered. The most natural would be that, if possible, the functional form of the altered query would be the same as that requested. Then a secondary objective would be that the support sets of the two queries be as close as possible, where closeness would best be measured from an application point of view. For instance, as indicated earlier, the support set corresponding to professors in a given department may be deemed close to that of professors in one of two related departments. For example, using the data from example (2), suppose that $S = \{1, 4\}$ and the users would like the answer to the query $q = q_3 = [1, 1, 0, 1]^T$. Since $q \in T$ and S is a maximal safe set from T , adding q to S will result in a disclosure. Instead, the DBM could try to find a safe query from the query set $C(q)$ given by those binary vectors that differ from q in, for example, exactly one coordinate. In other words,

$$C(q) := \left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right\}.$$

It is not difficult to see that there are only two queries in $C(q)$ that can be safely added to S : $[1, 0, 0, 1]^T$ and $[1, 1, 0, 0]^T$. The first one is already in S , so that a possible choice is to add $[1, 1, 0, 0]^T$. Summarizing, the DBM informs the user that she is unable to answer the original query q , but at least she can answer a close query corresponding to the sum of the records belonging to Subjects 1 and 2. The DBM could also have considered a set $C(q)$ defined as those binary vectors that differ from q in one or two coordinates; or considered a set $C(q)$ consisting of vectors that are not necessarily binary; or considered random queries generated by taking random samples of the subjects involved in a SUM query; etc.

The DBM can also offer the closest query in the set of linear combinations of queries in S , where closeness is measured in terms of an Euclidean norm. Again the user could do this herself if S is published with answers. In the case of linear queries, this approach leads to a simple least-squares problem. Let K be the index set of the queries in S , let $Q := Q_K$ denote the $n \times |K|$ matrix whose columns are the queries with indexes in S , and let q be a target query. Without loss of generality we assume that Q has full-column rank. Then, if q is not safe with respect to S and there is no query in the set $C(q)$ that can be safely added to S , we compute the least-squares approximation to q in the subspace of linear combinations of columns of Q . In other words, we compute

$$\Delta q := (QQ^+ - I)q,$$

where $^+$ denotes the standard pseudo-inverse (or Moore-Penrose inverse) operator of a matrix (see Golub and Van Loan (1997)), and answer $(q + \Delta q)^T a$.

Similarly, in the case of nonlinear queries the DBM can solve, in polynomial time, the least squares problem $\min\{\|\Delta q\| : q + \Delta q \text{ is a linear combination of } q_j, j \in K\}$, and then answer $(q + \Delta q)(a)$.

6. Computational Results

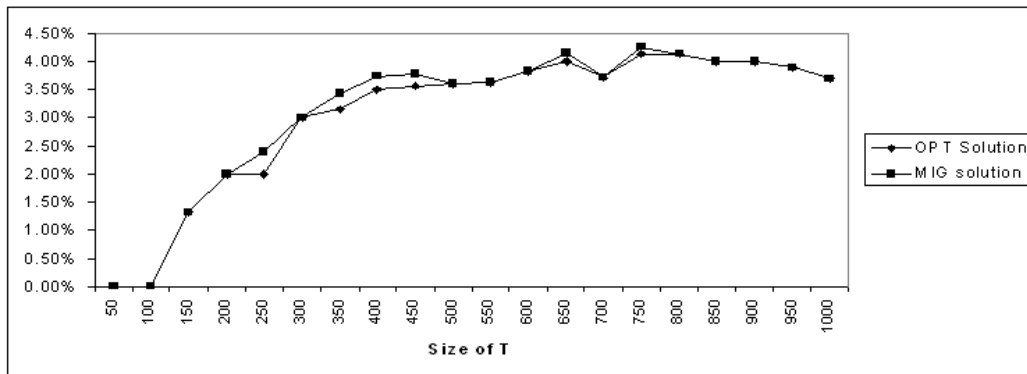
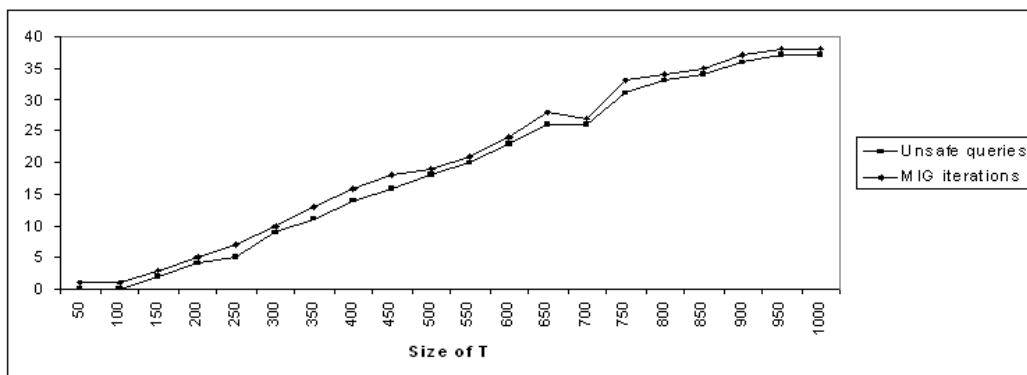
In this section we provide an experimental analysis of the proposed techniques. The experimental design consists of a database with $n = 1,000$ confidential values. The confidential vector a was uniformly randomly generated in the range $[10, 100]$ with an average of 55 and standard deviation of 26. The performance of our techniques is evaluated via 1,000 design time queries and 2,000 ad hoc run time queries. A mixture of linear and nonlinear queries were considered in both stages.

6.1. Design Stage Experiments

In the design stage 1,000 equally weighted queries, fifty percent of which are SUM queries and the remaining are VARIANCE queries, were considered. To be able to determine an optimal maximal subset we designed the sample so that it is possible to determine in polynomial time an optimal solution to any subset from the sample. The sample queries were generated as follows: we randomly selected 50 disjoint subsets N_1, \dots, N_{50} from the subject index set N , each subset with a cardinality between 5 and 20 elements. Next, we used the subsets to generate two sets of queries: simple and combined. The set of simple queries consists of 100 queries from taking the SUM and the VARIANCE queries over each of the 50 subsets. Notice that combining the exact answers to a SUM and a VARIANCE query over a given subset is not enough information to create a disclosure threat for the subjects in the subset (for example there are at least $5! = 120$ different ways to obtain the same given SUM and VARIANCE answers by permuting the confidential values of a subset). Further, since the subsets are disjoint, it is not possible to disclose a confidential value of a subset by releasing the exact answers to the queries from another subset. Therefore, the set of simple queries is a safe set, that is, there is no risk of disclosure by exactly answering any combination of simple queries.

To construct the set of combined queries, first we randomly chose one element k_j from each subset N_j . Next, for each index pair (i, j) with $i < j$, $i, j = 1, \dots, 50$, we created new subsets from N of the form $N_{i,j} := N_i \cup (N_j \setminus \{k_j\})$, where k_j is the randomly chosen element from N_j . Then, the set of combined queries consists of taking the SUM and VARIANCE queries over each of the combined $(49 \times 50)/2 = 1,225$ subsets, thus yielding a total of 2,450 combined queries. As with simple queries, the combination of the exact answers to a SUM and a VARIANCE query over a given combined subset is not enough information to create a disclosure threat for the subjects in the subset. Moreover, notice that the only disclosure threat that can arise from exactly answering combined queries would occur if the queries could be linearly combined to isolate the query answers comprising the subsets N_j and $N_j \setminus \{k_j\}$ (for example answering the SUM queries for N_j and $N_j \setminus \{k_j\}$ discloses the confidential value of subject k_j). Nevertheless, because every query always entails exactly two disjoint subsets, it is not possible to find a linear combination of combined queries to isolate the query answers comprising either N_j or $N_j \setminus \{k_j\}$. Therefore, the set of combined queries is also a safe set.

Given a nonempty target set T consisting of both simple and combined queries with equal weights, we wish to determine the optimal safe subset from T . Since all queries have the same weight, this is equivalent to finding a maximum cardinality safe subset from T . As we have observed, both sets of simple and combined queries are safe by themselves. However, when the queries are mixed, we can obtain an unsafe combination of queries, namely, when the SUM queries of the three subsets N_i , N_j , and $N_{i,j}$ are exactly answered together for $i < j$ (the confidential value of subject k_j in N_j will be disclosed). Hence, notice that providing an exact answer to a SUM query concerning a simple subset N_i can potentially compromise the safety of all SUM queries associated with combined subsets $N_{i,j}$ for $j > i$. On the other hand, providing an exact answer to a SUM query concerning a combined subset $N_{i,j}$ can potentially compromise the safety of the only two other SUM queries associated with the simple subsets N_i and N_j , respectively. Thus, if a conflict does arise involving SUM queries q_i, q_j , and $q_{i,j}$, $i < j$, in T for subsets N_i , N_j , and $N_{i,j}$, respectively, it is optimal to remove q_i or q_j depending on which is more conflicting in combined SUM queries across T . In other words, we compute how many SUM queries in T involving subsets of the form $N_{k,i}$ or $N_{i,k}$ conflict with q_i ;

Figure 3 Percentage of Rejected Queries Using OPT and MIG.**Figure 4** Major Iterations of MIG Algorithm.

how many SUM queries in T involving subsets of the form $N_{k,j}$ or $N_{j,k}$ conflict with q_j ; and then we remove q_i or q_j depending on which query produces the most conflicts.

The total number of safety conflicts in T does not exceed the total number of combined queries in T , and since resolving each conflict requires examination of at most all other queries in T , it is clear that identifying an optimal safe subset from T can be done in polynomial time (as a function of the cardinality of T) for the queries in our sample.

Our sample of 1,000 queries was obtained by randomly drawing 500 SUM queries and 500 VARIANCE queries from the union of simple and combined query sets. Next, we randomly shuffled the queries in the sample and determined 20 target sets of queries $T_1 \subset \dots \subset T_{20}$ with $|T_i| = 50i$. Set T_1 contains the first 50 queries in the random order, T_2 contains the first 100 queries in the random order, and so on. For each set T_i , we computed how many queries in T_i are exactly answered using an optimal solution to OPT and how many are exactly answered using the solution produced by algorithm MIG.

Figure 3 illustrates the relative performance of the two approaches, in terms of the percentage of target queries that were declined to be answered exactly, as a function of the set of the target query set. MIG and OPT produced identical results 60% of the time. For those sets in which the OPT and the MIG solutions are not the same, MIG always answered exactly one less query than the OPT solutions. Figure 4 shows the number of iterations required by algorithm MIG to determine a solution. The chart also shows the optimal number of unsafe queries for each target set. By keeping in mind that the MIG algorithm drops at most one unsafe query in each iteration, we observe that the algorithm follows the optimal pattern very closely.

Figure 5 Percentage of Rejected Queries Using Query Restriction and MIG.

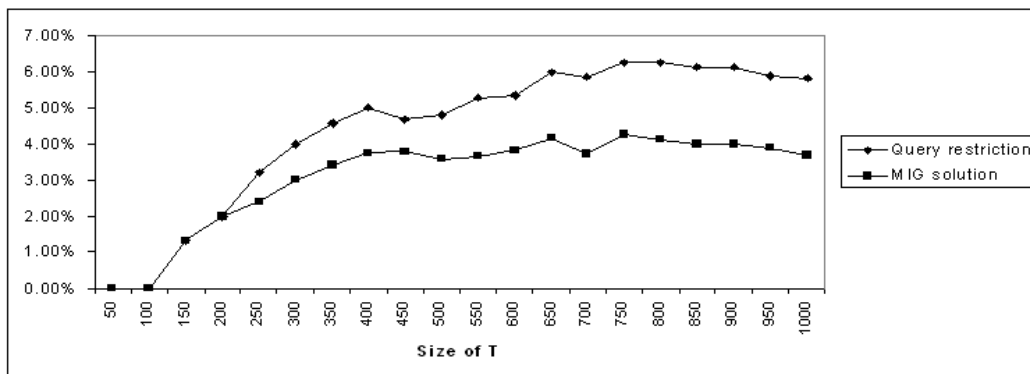


Figure 5 provides a comparative analysis of MIG with the query restriction approach. To implement query restriction, we process each of the queries in the sample according to the established random sequence order. In each iteration, we check to see if the query is safe with respect to those already answered. If it is, we add it to the set of answered queries. Otherwise the query request is denied. Figure 5 shows that the query restriction approach rejects approximately twice the number of queries not answered exactly by MIG.

These results indicate that the performance of MIG is near optimal and that MIG significantly outperforms the current query restriction techniques.

6.2. Run Time Stage Experiments

The run time stage of the experimental analysis consisted of 2,000 randomly generated ad hoc queries of four types: sum, variance, max, and median. Two hundred and fifty queries of each type were used in the analysis. The size of each query was uniformly randomly generated from the range [5, 1000]. Given a query of size k , we sampled without replacement k numbers from $\{1, \dots, 1000\}$ to determine the set of confidential entries from the database associated with the query.

The performance of the proposed techniques was evaluated using a standard perturbation approach as the benchmark. A standard perturbation vector Δa was created with each entry randomly generated according to a normal distribution with mean zero and standard deviation of 15. Then we projected the resulting perturbed vector $a + \Delta a$ as described in Section 4 to obtain three additional perturbed vectors $a + \Delta a_1$, $a + \Delta a_2$, and $a + \Delta a_3$; each vector consistent with the answers to the queries in the OPT sets computed from target sets T_1 , T_{10} , and T_{20} in the design stage, respectively. Finally, for each query q in the sample we evaluated the query using each perturbed vector and computed the relative error with respect to the original data

$$\text{Rel}(q) := \frac{|q(a + \Delta x) - q(a)|}{|q(a)|},$$

when $q(a) \neq 0$, where Δx is the corresponding perturbation. Table 5 summarizes the results from the run time stage simulation.

The results clearly indicate that the proposed approach provides more accurate query answers than the standard perturbation approach. This follows intuitively, since the perturbed vector is constrained to have additional characteristics in common with the original data set beyond simple addition of random noise. That is, the original and the perturbed vector lie on the same hyperplane determined by the exact answers of queries in set S . Also, in the likely case that run-time queries are “similar” to those in set S , these performance benefits are expected to be further enhanced. Interestingly the performance enhancements were quite significant for sum and median queries. This

Table 5 Improvement of Hybrid Perturbation over Standard Perturbation.

Perturbation	Query Type				
	Sum	Variance	Max	Median	All
$a + \Delta a_1$	25.8%	1.4%	3.7%	19.7%	4.4%
$a + \Delta a_2$	23.8%	2.2%	3.8%	20.1%	4.7%
$a + \Delta a_3$	21.3%	2.3%	3.8%	23.8%	5.0%

follows logically from the additive nature of the perturbation technique. In summary these results highlight the practical benefits that can accrue from implementing the proposed approach.

7. Conclusions and Further Work

We have shown that it is possible to efficiently implement a hybrid protection technique that provides exact answers to a great number of target queries and provides perturbed answers to all other queries. Our technique incorporates the advantages of the query restriction and standard perturbation techniques. We also show that our approach applies to linear and nonlinear queries and to numerical and categorical data.

Our work can be extended to provide protection against insider threats. Insider threat generally occurs when a database user knows one or more confidential values. A user could combine this knowledge with the released answers to infer the values of other confidential data. A typical example is a user who also is a subject of the database and at least knows her own confidential datum. Insider threats can be controlled by adding query-set size constraints to the query target set, or by using a probabilistic model to model insider knowledge. For instance, the DBM could eliminate from consideration those queries in the target set with sizes outside a predetermined range. Alternatively, the DBM could try to estimate the probability that a randomly chosen user will know a given set of confidential values. As suggested in Fienberg et al. (1997), the DBM can determine an a-priori probability distribution on the user’s knowledge and then can compute the probability of having at least one disclosure resulting from the exact answers to a set of queries. In either case, finding an efficient way to integrate protection against this type of threats to our hybrid technique is a relevant extension.

Appendix

A. Proof of Results

Proof of Theorem 1: First, suppose that $K \in \mathcal{F}_i$ and let $R \subset K$. Let $C_K := \{q_j : j \in K\}$ and $C_R := \{q_j : j \in R\}$. Without loss of generality we assume the the queries in C_K are linearly independent. By definition, $C_K \cup \{e_i\}$ is a collection of linearly independent vectors in \mathfrak{R}^n . Since, $C_R \cup \{e_i\} \subset C_K \cup \{e_i\}$, it follows that $C_R \cup \{e_i\}$ is also a collection of linearly independent vectors, and so, e_i is not a linear combination of the queries with indexes in R . Therefore, $R \in \mathcal{F}_i$.

Second, let R and K be sets in \mathcal{F}_i such that $|K| = |R| + 1$. As before, without loss of generality we assume that the queries in C_K and C_R are linearly independent. Suppose that $C_R \cup \{q_j\} \cup \{e_i\}$ is a collection of linearly dependent vectors for all $j \in K$ such that $q_j \in C_K \setminus C_R$. Since by definition, $C_R \cup \{e_i\}$ is a collection of linearly independent vectors, it follows that q_j is a linear combination of the elements of $C_R \cup \{e_i\}$ for all $j \in K$ such that $q_j \in C_K \setminus C_R$. Hence, every vector in C_K is in C_R or is a linear combination of the elements of $C_R \cup \{e_i\}$. Therefore, $\dim(C_K \cup \{e_i\}) \leq |R| + 1$. On the other hand, since by definition $C_K \cup \{e_i\}$ is a collection of linearly independent vectors, we have $\dim(C_K \cup \{e_i\}) = |K| + 1 = |R| + 2$, a contradiction. Therefore, there exists $q_j \in C_K \setminus C_R$ such that $C_R \cup \{q_j\} \cup \{e_i\}$ is a collection of linearly independent vectors, and so, there exists $j \in K \setminus R$ such that $R \cup \{j\} \in \mathcal{F}_i$.

It follows from Theorem 5.1 on page 273 from Lawler (2001) that all maximal sets in \mathcal{F}_i have the same cardinality, and so, (M, \mathcal{F}_i) is a matroid. ■

Proof of Proposition 1: Given a subset K of M , let Q_K denote the matrix whose columns are the query vectors q_j with $j \in K$. Since we are assuming that the queries with index set K are linearly independent, it follows that

$$\text{rank}(Q_K) = |K| \leq n,$$

that is, Q_K has full column rank. Moreover, K is a safe set with respect to index i if and only if the system

$$Q_K x = e_i, \tag{13}$$

is infeasible. Let k denote the cardinality of K . Consider the *LUP* decomposition of the matrix Q_K , that is,

$$P_K Q_K = L_K U_K, \tag{14}$$

where U_K is a $k \times k$ upper-triangular matrix, P_K is an $n \times n$ permutation matrix, and L_K is an $n \times k$ matrix of the form

$$L_K = \begin{bmatrix} L_0 \\ L_1 \end{bmatrix},$$

with L_0 a $k \times k$ unit lower-triangular matrix. Since Q_K has full column rank, it follows that U_K is a non-singular matrix, and so, using (13) and (14) we have that K is a safe set with respect to index i if and only if the system

$$L_K y = P e_i = e_{\pi(i)}, \tag{15}$$

is infeasible, where $e_{\pi(i)}$ is the canonical vector resulting from applying the permutation matrix P to e_i . Because of the structure of the matrix L_K , determining the feasibility of system (15) can be done by using forward substitution on the submatrix L_0 to find y and then testing if $L_1 y$ agrees with the lower $n - k$ dimensional subvector of $e_{\pi(i)}$. Hence, if the *LUP* decomposition of the matrix Q_K is known, determining if K is a safe set with respect to index i will take $O(kn)$ arithmetical operations.

On the other hand, computing the *LUP* decomposition of an arbitrary $n \times k$ dimensional matrix A , $n \geq k$, takes $O(k^2 n)$ arithmetical operations. Moreover, updating the decomposition of A , after adding or deleting a column from A takes $O(kn)$ arithmetical operations. Since \mathcal{F}_i is a matroid, we can use the greedy algorithm to compute the rank of a subset $K \subset M$. We start with an arbitrary query with index in K and create an initial matrix with only one column corresponding to the chosen query. Then we keep adding queries (columns) to the matrix as long as we obtain independent (safe) sets with respect to i . When adding or deleting a column we update the current *LUP* decomposition of the test matrix. Therefore, computing $r_i(K)$ takes $O(k^2 n)$ arithmetical operations.

It follows that $r_i^D(K)$ can be computed in no more than $O(nm^2)$ operations for all $1 \leq i \leq n$, and so, computing $f(K)$ takes no more than $O(m^2 n^2)$ operations. Finally, the most computationally demanding step in MIG is the step

$$j_t \leftarrow \text{argmin}\{w_j / (f(K^{t-1} \cup \{j\}) - f(K^{t-1})) : j \in M^t\},$$

which requires no more than $O(m^3 n^2)$ operations. Since, that step is executed at most m times, we get our bound of $O(m^4 n^2)$ arithmetical operations. ■

Proof of Proposition 2: Let \mathcal{H} denote a Hilbert space, that is, an inner product complete vector space (Luenberger 1968) with dimension n or greater and such that $T \subset \mathcal{H}$ and $\pi_i \in \mathcal{H}$ for all i . We denote by $(\cdot | \cdot)$ the inner product of \mathcal{H} , and by $\|q\| := (q|q)^{1/2}$, $q \in \mathcal{H}$, the norm induced by the inner product. For a subset $K \subset M$, we denote by $G(K)$ the Gram matrix generated by the queries $\{q_j : j \in K\}$, that is, every entry of $G(K)$ is of the form $(q_i | q_j)$ with $i, j \in K$. Notice that $G(K)$ is

a $|K| \times |K|$ symmetric matrix. If q is an arbitrary query, let $v_K(q)$ be the $|K|$ dimensional vector whose coordinates are of the form $(q|q_j)$ with $j \in K$. Then, using Theorem 1 from Luenberger (1968, page 57), if the queries in the set $\{q_j : j \in K\}$ are linearly independent, then

$$\delta(q, K) := \|q - \hat{q}\|, \quad (16)$$

is the minimum distance from the query q to the subspace $\langle q_j : j \in K \rangle$, where \hat{q} satisfies

$$\hat{q} = \sum_{j \in K} \hat{x}_j q_j, \quad (17)$$

and \hat{x} is the (vector) solution to the *normal* equations:

$$G(K)x = v_K(q). \quad (18)$$

The relevance of (16)-(18) is that it provides a test of whether or not an arbitrary query is a linear combination of a subset of queries, namely, whether or not $\delta(q, K) = 0$. This follows from noticing that $\langle q_j : j \in K \rangle$ is a closed set because it is a finite-dimensional subspace of \mathcal{H} , and so, $\delta(q, K) = 0$ if and only if $q \in \langle q_j : j \in K \rangle$. More importantly, (16)-(18) show that the test is equivalent to solving a linear system of equations, which can be done efficiently once the Gram matrix $G(K)$ and the vector $v_K(q)$ in (18) are computed.

Therefore, the complexity of implementing step (7) from Algorithm 3 by using the Hilbert space approach reduces to defining an adequate inner product (and corresponding space) to allow for an efficient computation of the parameters in the normal equations. To do so, consider a finite set of linearly independent, randomly generated, n -dimensional vectors

$$Y := \{y_1, \dots, y_s\} \subset \mathfrak{R}^n, \quad (19)$$

where $\min\{n, t\} \leq s \leq n$ and such that the queries in T are defined at each point in Y . The space defined as $\mathcal{H} := \mathfrak{R}^Y$, that is, the space of queries mapping Y into \mathfrak{R} , with inner product is given by

$$(q_1|q_2) := \sum_{i=1}^s q_1(y_i)q_2(y_i). \quad (20)$$

It is easy to check that \mathcal{H} is a Hilbert space. Also notice that the Gram matrix associated with a subset $\{q_1, \dots, q_t\}$ of queries satisfies

$$\begin{aligned} G(q_1, \dots, q_t) &= \begin{bmatrix} (q_1|q_1) & (q_1|q_2) & \cdots & (q_1|q_t) \\ (q_2|q_1) & & & \\ \vdots & & & \vdots \\ (q_t|q_1) & & \cdots & (q_t|q_t) \end{bmatrix} \\ &= \begin{bmatrix} q_1(y_1) & q_2(y_1) & \cdots & q_t(y_1) \\ q_1(y_2) & & & \\ \vdots & & & \vdots \\ q_1(y_s) & & \cdots & q_t(y_s) \end{bmatrix}^T \begin{bmatrix} q_1(y_1) & q_2(y_1) & \cdots & q_t(y_1) \\ q_1(y_2) & & & \\ \vdots & & & \vdots \\ q_1(y_s) & & \cdots & q_t(y_s) \end{bmatrix}. \end{aligned}$$

Hence, if for a given subset $K \subset M$ we denote by

$$Y(K) := \begin{bmatrix} q_1(y_1) & q_2(y_1) & \cdots & q_t(y_1) \\ q_1(y_2) & & & \\ \vdots & & & \vdots \\ q_1(y_s) & & \cdots & q_t(y_s) \end{bmatrix},$$

the matrix whose columns are the queries with indexes in K evaluated at each point in Y , we obtain

$$G(K) = Y(K)^T Y(K),$$

and

$$v_K(q) = Y(K)^T \begin{bmatrix} q(y_1) \\ q(y_2) \\ \vdots \\ q(y_s) \end{bmatrix}.$$

When the points in Y are chosen so that the matrix $Y(K)$ has full-column rank, $G(K)$ is nonsingular and the solution to the normal equations can be efficiently found using a standard linear system numerical method.

In conclusion, from the previous discussion and using similar arguments from the proof of Proposition 1, we obtain the result. ■

B. Example of Algorithm MIG Applied to Nonlinear Queries

Consider a 4-dimensional database vector of the form $a := [50, 50, 60, 70]^T$, and the set of target queries $T := \{q_1, q_2, q_3, q_4, q_5\}$, where the queries are defined as

$$\begin{aligned} q_1(x) &:= x_1 + x_2 + x_3 + x_4; \\ q_2(x) &:= x_1 + x_2 + x_3; \\ q_3(x) &:= \text{VAR}(x_1, x_2, x_3, x_4); \\ q_4(x) &:= q_1(x) + q_3(x); \\ q_5(x) &:= q_2(x) + q_3(x). \end{aligned}$$

To compute a Gram matrix associated with any set of queries, we use the following sample set Y ,

$$Y := \{y_1, y_2, y_3, y_4\} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Clearly, the elements of Y are linearly independent.

To illustrate how a Gram matrix is computed, consider the query index set $K := \{1, 2, 3\}$. The matrix $Y(K)$ consisting of evaluations of the queries in K on the sample points is

$$Y(K) = \begin{bmatrix} q_1(y_1) & q_2(y_1) & q_3(y_1) \\ q_1(y_2) & q_2(y_2) & q_3(y_2) \\ q_1(y_3) & q_2(y_3) & q_3(y_3) \\ q_1(y_4) & q_2(y_4) & q_3(y_4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3/16 \\ 2 & 2 & 4/16 \\ 3 & 3 & 3/16 \\ 4 & 3 & 0 \end{bmatrix}.$$

Hence, the corresponding Gram matrix $G(K)$ is

$$G(K) = Y(K)^T Y(K) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 3 \\ 3/16 & 4/16 & 3/16 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 3/16 \\ 2 & 2 & 4/16 \\ 3 & 3 & 3/16 \\ 4 & 3 & 0 \end{bmatrix} = \begin{bmatrix} 30 & 26 & 5/4 \\ 26 & 23 & 5/4 \\ 5/4 & 5/4 & 34/256 \end{bmatrix}.$$

Now we show that the set K is unsafe. Consider the projection query π_4 . We first solve the system

$$G(K)x = v_K(\pi_4),$$

Table 6 Results from the Approximation Algorithm.

t	M^t	j_t	K^t	$f(K^t)$	$W(M \setminus K^t)$
0	-	-	\emptyset	0	21
1	{1, 2, 3, 4, 5}	2	{2}	4	18
2	{1, 3, 4, 5}	4	{2, 4}	8	14
3	{1, 3, 5}	5	{2, 4, 5}	9	10

where

$$v_K(\pi_4) = Y(K)^T \begin{bmatrix} \pi_4(y_1) \\ \pi_4(y_2) \\ \pi_4(y_3) \\ \pi_4(y_4) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 3 \\ 3/16 & 4/16 & 3/16 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 0 \end{bmatrix}.$$

By solving the system, we obtain

$$\hat{x} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}.$$

Next, we take a linear combination of the columns of $Y(K)$ according to the vector \hat{x} , that is, we compute

$$Y(K)\hat{x} = \begin{bmatrix} 1 & 1 & 3/16 \\ 2 & 2 & 4/16 \\ 3 & 3 & 3/16 \\ 4 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

which is identical to $[\pi_4(y_1), \pi_4(y_2), \pi_4(y_3), \pi_4(y_4)]^T$. Therefore, $\delta(\pi_4, K) = 0$ and the method correctly identifies that π_4 is a linear combination of the queries indexed by K , and so, K is not a safe set.

To illustrate how Algorithm 3 operates on K to compute the rank function value $r_4(K)$, notice that in the first iteration an index, say $k = 1$, is removed from K . Then, since $R = \emptyset$ at this point, we have $q_1 \notin \langle q_j : j \in R \rangle = \emptyset$ and $\pi_4 \notin \langle q_j : j \in R \cup \{1\} \rangle = \langle q_1 \rangle$. Hence, index 1 is added to R . In the second iteration, another index, say $k = 2$, is removed from K . In this case, we have $q_2 \notin \langle q_j : j \in R \rangle = \langle q_1 \rangle$ (q_1 and q_2 are linearly independent), but $\pi_4 \in \langle q_j : j \in R \cup \{2\} \rangle = \langle q_1, q_2 \rangle$ because the corresponding solution of the Gram system is $\hat{x} = [1, -1]^T$ and $\pi_4 = q_1 - q_2$. Hence, index 2 is not added to R . Finally, the last index, $k = 3$, is removed from K . We have $q_3 \notin \langle q_j : j \in R \rangle = \langle q_1 \rangle$ (q_1 and q_3 are also linearly independent), and $\pi_4 \notin \langle q_j : j \in R \cup \{3\} \rangle = \langle q_1, q_3 \rangle$ because the corresponding solution of the Gram system is $\hat{x} = [34/155, -64/31]^T$ and $\pi_4 \neq (34/155)q_1 - (64/31)q_3$. Therefore, the algorithm concludes that $r_4(K) = 2$.

Now, we illustrate how the MIG algorithm operates on the queries in T . Let $M := \{1, 2, 3, 4, 5\}$ be the query index set corresponding to T . Assign weights of $w_1 = 5$, $w_2 = 3$, $w_3 = 5$, $w_4 = 4$, and $w_5 = 4$ to the queries in T , respectively.

In Table 6, we show the result of each iteration of algorithm MIG. Notice that since $f(M) = 9$, the algorithm finishes at iteration $t = 3$. The set returned by the algorithm is $\hat{K} = \{2, 4, 5\}$, and so, the associated set $M \setminus \hat{K} = \{1, 3\}$ is safe. Moreover, it is not hard to see that $\{1, 3\}$ is also optimal, that is, the solution provided by MIG is the optimal solution.

Next, we compute a perturbation that preserves the answers to the optimal set of queries. For our database $a = [50, 50, 60, 70]^T$ we obtain $q_1(a) = 230$ and $q_3(a) = 68.75$. A perturbation Δa that preserves the same answers must satisfy equations $q_1(a + \Delta a) = q_1(a)$ and $q_{13}(a + \Delta a) = q_3(a)$, which lead to the equations

$$\begin{aligned} \Delta a_1 + \Delta a_2 + \Delta a_3 + \Delta a_4 &= 0, \\ \Delta a_1^2 + \Delta a_2^2 + \Delta a_3^2 + \Delta a_4^2 + 2(50\Delta a_1 + 50\Delta a_2 + 60\Delta a_3 + 70\Delta a_4) &= 0. \end{aligned}$$

There are an infinite number of solutions to those equations. In particular, $\Delta a = [5, -5, 5, -5]^T$ is a solution, thus yielding the perturbed database $a + \Delta a = [55, 45, 65, 65]^T$. It is easy to verify that $q_1(a + \Delta a) = 230$ and $q_3(a + \Delta a) = 68.75$.

Finally, we provide answers to the other queries in T . To answer q_2 and q_5 we use the perturbed database to obtain $q_2(a + \Delta a) = 165$ and $q_5(a + \Delta a) = 233.75$. Clearly, since q_4 is a linear combination of q_1 and q_3 , we have $q_4(a + \Delta a) = 298.75 = q_4(a)$, which is an exact answer.

Acknowledgments

The authors would like to thank the Associate Editor and anonymous referees for their careful reading and helpful suggestions that significantly improved this work. The second and third authors wish to acknowledge the support received from TECI - The Treibick Electronic Commerce Initiative, Department of Operations and Information Management, University of Connecticut.

References

- Abul-Ela, A.-L., B.G. Greenberg, D.G. Horvitz. 1967. A multi-proportions randomized response model. *Journal of the American Statistical Association* **62**(319) 990–1008.
- Bazaraa, M.S., H.D. Sherali, C.M. Shetty. 1993. *Nonlinear Programming, Theory and Algorithms*. 2nd ed. John Wiley & Sons, Inc., New York.
- Castano, S., M. Fugini, G. Martella, P. Samarati. 1996. *Database Security*. Addison-Wesley Publishing Company, Reading, MA.
- Chin, F., G. Ozsoyoglu. 1982. Auditing and inference control in statistical databases. *IEEE Transactions On Software Engineering* **SE-8**(6) 574–582.
- Cormen, T., C. Leiserson, R. Rivest. 2000. *Introduction to Algorithms*. The MIT Press, Cambridge, MA.
- Cox, L.H. 1980. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association* **75**(370) 377–385.
- Denning, D.E. 1982. *Cryptography and Data Security*. Addison-Wesley Publishing Company, Reading, MA.
- Duncan, G.T., S. Mukherjee. 2000. Optimal disclosure limitation strategy in statistical databases: Detering tracker attacks through additive noise. *Journal of the American Statistical Association* **95**(451) 720–729.
- Fienberg, S.E., U.E. Makov, A.P. Sanil. 1997. A Bayesian approach to data disclosure: Optimal intruder behavior for continuous data. *Journal of Official Statistics* **3**(1) 75–89.
- Fischetti, M., J.J. Salazar. 1999. Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control. *Mathematical Programming* **84**(2) 283–312.
- Golub, G.H., C.F. Van Loan. 1997. *Matrix Computations*. 3rd ed. The Johns Hopkins University Press, New York.
- Gopal, R., P. Goes, R. Garfinkel. 1998. Interval protection of confidential information in a database. *Journal On Computing* **10**(3) 309–322.
- Lawler, E. 2001. *Combinatorial Optimization, Networks and Matroids*. Dover Publications, Inc., New York.
- Lehmann, E.L. 1998. *Elements of Large-Sample Theory*. Springer-Verlag, New York.
- Luenberger, D.G. 1968. *Optimization by Vector Space Methods*. John Wiley & Sons, Inc., New York.
- Muralidhar, K., D. Batra, P.J. Kirs. 1995. Accesibility, security, and accuracy in statistical databases: The case for the multiplicative fixed data perturbation approach. *Management Science* **41**(9) 1549–1564.
- Muralidhar, K., R. Parsa, R. Sarathy. 1999. A general additive data perturbation method for database security. *Management Science* **45**(10) 1399–1415.
- Muralidhar, K., R. Sarathy, R. Parsa. 2001. An improved security requirement for data perturbation with implications for e-commerce. *Decision Sciences* **32**(4) 683–698.

- Nemhauser, G.L., L.A. Wolsey. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research* **3**(3) 177–188.
- Nemhauser, G.L., L.A. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., New York.
- Nemhauser, G.L., L.A. Wolsey, M.L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming* **14** 265–294.
- Statistics New Zealand. 2003. *2001 Census of Population and Dwellings - Meshblock Database Content*. Government of New Zealand. Available at:
www.stats.govt.nz/NR/rdonlyres/3C972F72-2CF2-4CAA-A7A0-015D76A72CD6/0/mbdbcontent.xls.
- Traub, J.F., Y. Yemini, H. Wozniakowski. 1984. The statistical security of a statistical database. *ACM Transactions On Database Systems* **9**(4) 672–679.
- Warner, S.L. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association* **60**(309) 63–69.