International Journal of Computer and Electrical Engineering, Vol. 1, No. 3, August 2009
1793-8163

Efficient Algorithms for the Key Representation Auditing Scheme

Asim A. Elshiekh, and P. D. D. Dominic

***Abstract*—A statistical database (SDB) publishes statistical queries (such as sum, average, count, etc) on subsets of records. Sometimes by stitching the answers of some statistics, a malicious user (snooper) may be able to deduce confidential information about some individuals. The key representation auditing scheme is proposed to guarantee the security of online and dynamic SDBs. The core idea is to convert the original database into key representation database (KRDB), also this scheme involves converting each new user query from string representation into key representation query (KRQ), and stores it in the Audit Query table (AQ table). Three audit stages are proposed to repel the attacks of the snooper to the confidentiality of the individuals. In this paper, efficient algorithms for these stages are presented, namely the First Stage Algorithm (FSA), the Second Stage Algorithm (SSA), and the Third Stage Algorithm (TSA). These algorithms enable the key representation auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB. Also, cost estimation for this scheme is performed, and we illustrate the saving in block accesses (CPU time) and storage space that are attainable when a KRDB is used.**

***Index Terms*—Auditing, compromise, confidentiality, statistical database.**

I. INTRODUCTION

A statistical database (SDB) is a database that is used for statistical queries (for example, SUM, AVERAGE, COUNT, etc) on subsets of the database entities [1]. Many government agencies, businesses, and nonprofit organizations need to collect, analyze, and report data about individuals in order to support their planning activities. SDBs therefore contain confidential information such as income, credit ratings, type of disease, or test scores of individuals. Such data are typically stored online and analyzed using sophisticated database management systems (DBMSs). On one hand, such database systems are expected to satisfy user requests of aggregate statistics related to non confidential and confidential attributes. On the other hand, the system should be secure enough to guard against the ability of a malicious user (snooper) to infer any confidential information about any individual represented in the database [2].

Manuscript received April 29, 2009.

Asim. A. Elshiekh is with the Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Seri Iskandar, Perak, Malaysia.

P. D. D. Dominic is with the Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Seri Iskandar, Perak, Malaysia.

Protecting an SDB means preventing and avoiding statistical inference. Inference in SDB means the possibility of obtaining confidential information on single entity, by taking advantage of (sequences of) statistical queries issued against a set of entities stored in the SDB. When confidential information about individuals is obtained, the database is said to be disclosed [1].

SDBs may be online or offline. In an online SDB, users get real-time responses to their statistical queries. Whereas, in an offline SDB, users do not know when their statistics will be processed, making disclosure more difficult. Also, SDBs may be static or dynamic. A static SDBs do not change during their lifetime (namely, no insertion or deletion operations occur), and possible changes give rise to new static database. In contrast, dynamic SDBs can change continuously. Protecting a dynamic SDB is more complex, since variations in the database state provide additional information to snoopers (malicious users) [1][2].

Hence, this paper will introduce a new scheme to protect online and dynamic SDBs from being disclosed. This scheme can guarantee the security of online and dynamic SDBs, provide precise and accurate responses, besides it needs less CPU time and storage space during query processing.

A. Overview of Solution Approaches

There are many inference control methods proposed to protect various database systems. Those methods for SDBs can be classified under three general approaches: data perturbation, output perturbation, and query restriction. Data perturbation approach introduces noise in the data. The original SDB is typically transformed into a modified (perturbed) SDB, which is then made available to researchers. The output perturbation approach perturbs the answer to user queries while leaving the data in the SDB unchanged. While, query restriction approach imposes extra restriction on queries which includes query-set-size control, query-set-overlap control, auditing, cell suppression, and partitioning [2]-[4].

Auditing of an SDB involves keeping up-to-date logs of all queries made by each user and constantly checking for possible compromise whenever a new query is issued. Auditing has the advantages such as allowing the SDB to provide users with unperturbed response that will not make disclosed. It has long believed that auditing is an effective tool for protection [5].

B. Attribute Classification

Each attribute can be classified into two types: category and data attributes. The category attributes are used to

identify and select records, and each of it contains specific domain. For example, assume that the domains of the category attributes Gender and Dept be {M, F} and {CS, EE, ME, CE, PE}, respectively [6][7]. While the data attributes hold other information, usually numerical, for which some statistical queries may be desired such as Salary, Score, Income, etc [6].

II. Previous Work

The problems of SDB security have been of growing concern in recent years [8]. A practical approach based on auditing was proposed in [9], called Audit Expert. The approach maintains a matrix to audit the history of user's queries and detects all of the possible disclosed conditions. The audit expert maintains a binary matrix whose columns represent database entities and whose rows represent the user queries that have already been answered. When a new query is issued, the matrix is updated. A row with all zeros except for an ith column indicates that exact disclosure of the confidential attribute of the corresponding entity is possible. Thus, the answer to the new query should be denied. The authors in [3][4] proposed an algorithm to enhance Chin's scheme [9] and reduce its time and space requirements. With the proposed method, Chin's scheme can be extended so that it can be used in dynamic SDBs. They also proposed an audit scheme for dynamic SDBs which requires less time and storage requirements, and does not have the space explosion problem that appears in Chin's scheme. In [10], the author proposed an auditing method. It is based on the offline auditing subcube queries model used in [11], which views an SDB as a function f from strings of k bits to the positive and negative integers with the keys being the domain of f. A query is always of length k bits; for example, for k = 5 a possible query could be l**0*, with s 0's and l's (in this case s = 2) and the * standing for "do not care". In [12], the authors presented an implementation of the auditing strategy to avoid both exact and approximate disclosure. The key data structure of their study is a query map, which is a graphical summary of answered queries. Since the size of a query map may be exponential in the number of answered queries, they introduced a query restriction criterion to make every query map a graph. Also they presented an auditing procedure on such a graph and discussed the computational issues connected with its implementation. The authors in [7] focused on sum-queries with a response variable of nonnegative real type and they proposed a compact representation of answered sum-queries, called an information model in "normal form", which allows the query system to decide whether the value of a new sum-query can or cannot be safely answered. If it cannot, then the query system will issue the range of feasible values of the new sum-query consistent with previously answered sum-queries. In [13], the authors studied the Boolean auditing problem for offline SDBs, where the data elements are Boolean, and the queries are sum queries over the integers. They proved certain complexity results suggesting that there is no general efficient solution for the auditing problem in this case. The authors in [14] considered the online query auditing. They illustrated how denials that depend on the answer to the current query can leak information and introduced the notion of simulatablility to tackle this problem. They provided simulatable algorithms for auditing sum queries and max queries under classical disclosure. In addition, they introduced a probabilistic notion of disclosure and provided an algorithm for auditing sum queries over real-valued data drawn uniformly from a bounded range under this notion. In [15], the authors considered the online query auditing problem. They constructed auditors for max queries and bags of max and min queries in both the partial and full disclosure settings. Their algorithm for the partial disclosure setting involves a novel application of probabilistic inference techniques. The authors in [16] provided an offline auditing framework for determining whether a database system adheres to its data disclosure policies. The auditor detects queries that accessed sensitive data by formulating an audit expression that declaratively specifies sensitive table cells.

Query auditing is an effective strategy for guarding the confidentiality of individual in statistical database [12], that is because it provide users with precise and accurate answers (unperturbed responses). Since a detailed examination of the inference problem reveals that we are have not yet arrived at a general and acceptable solution [17], a new auditing scheme is proposed which can guarantee the security of online and dynamic SDBs, provide precise and accurate responses, besides it needs less CPU time and storage space during query processing.

III. Key Representation Auditing Scheme for Online and Dynamic SDBs

A statistical database (SDB) is said to be secure if no confidential data about any individual can be inferred from the available queries. By auditing, all user queries are logged and checked for possible inference before the result of the new query is released. In this section, a new auditing scheme is proposed to protect online and dynamic SDBs from being disclosed, also to prevent each new user query that could lead to disclose the SDB.

A. Statistical Database Model

Assume that the original database $\boldsymbol{D}$, which in both string and numerical representations, contains $\boldsymbol{N}$ records of individuals. Each record has $\boldsymbol{t}$ category attributes and $\boldsymbol{d}$ data attributes ($\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_t, \mathbf{A}_{t+1}, \mathbf{A}_{t+2}, \ldots, \mathbf{A}_{t+d}$).

Each category attribute $\mathbf{A}_j$ ($1 \le j \le t$) has $|\mathbf{A}_j|$ possible values, namely the domain of each category attribute has $|\mathbf{A}_j|$ classes. For example, the category attribute Gender, whose two possible values (or classes) are Male and Female. Let $\mathbf{\Theta}_{ikj}$ be the domain of the category attribute $\mathbf{A}_j$ ($1 \le i \le N$; $1 \le k \le |\mathbf{A}_j|$; $1 \le j \le t$), where the subsets $\boldsymbol{i}$, $\boldsymbol{k}$, and $\boldsymbol{j}$ represent the record number, the category attribute class, and the attribute number, respectively. Thus, the domains of the $\boldsymbol{t}$ category attributes are as following:

$$\mathbf{A}_1 = \{\mathbf{\Theta}_{i11}, \mathbf{\Theta}_{i21}, \ldots, \mathbf{\Theta}_{i|\mathbf{A}_1|1}\}$$
$$\mathbf{A}_2 = \{\mathbf{\Theta}_{i12}, \mathbf{\Theta}_{i22}, \ldots, \mathbf{\Theta}_{i|\mathbf{A}_2|2}\}$$
$$\vdots$$
$$\mathbf{A}_j = \{\mathbf{\Theta}_{i1j}, \mathbf{\Theta}_{i2j}, \ldots, \mathbf{\Theta}_{i|\mathbf{A}_j|j}\}$$
$$\vdots$$

$$\mathbf{A}_t = \{\mathbf{\Theta}_{i1t}, \mathbf{\Theta}_{i2t}, \ldots, \mathbf{\Theta}_{i|\mathbf{A}_t|t}\}$$

While each data attribute $\mathbf{A}_j$ ($t+1 \le j \le t+d$) hold other information, usually numerical, for which some statistical queries may be desired such as Salary, Score, Income, etc. Let $\mathbf{V}_{ij}$ ($1 \le i \le N$; $1 \le j \le d$) be the values of the d data attributes ($\mathbf{A}_{t+1}, \mathbf{A}_{t+2}, \ldots, \mathbf{A}_{t+d}$). Thus, the values of the d data attributes are as following:

$$\mathbf{V}_{i1}, \mathbf{V}_{i2}, \ldots, \mathbf{V}_{ij}, \ldots, \mathbf{V}_{id}$$

B. Key Representation Database (KRDB)

The core idea of this work is to convert the original database $\boldsymbol{D}$ into key representation database $\boldsymbol{D}^{\backslash}$, which consists of ($1+d$) cells. The t category attributes in the original database $\boldsymbol{D}$ will be converted into one cell, and the d data attributes will be separated by the sign '**.**'. Each category attribute value $\mathbf{\Theta}_{ikj}$ ($1 \le i \le N$; $1 \le k \le |\mathbf{A}_j|$; $1 \le j \le t$) in the original database $\boldsymbol{D}$ will be replaced by its category attribute class (k=1, 2, …, or $|\mathbf{A}_j|$). The converted ($1+d$) cells are as following:

$$\mathbf{U}_{i1}\mathbf{U}_{i2}\ldots\mathbf{U}_{it}\mathbf{.V}_{i1}\mathbf{.V}_{i2}\mathbf{.}\ldots\mathbf{.V}_{id}$$

Where, $\mathbf{U}_{ij}$ ($1 \le i \le N$; $1 \le j \le t$) represents the category attribute class corresponding to the category attribute $\mathbf{A}_j$ for the record number i. And $\mathbf{V}_{ij}$ ($1 \le i \le N$; $1 \le j \le d$) represents the d data attributes for the record number i separated by the sign '**.**'.

For example, let t = 4 and d = 2, also assume the following information describe the record number i in the original database $\boldsymbol{D}$:

($\mathbf{\Theta}_{i31}$, $\mathbf{\Theta}_{i12}$, $\mathbf{\Theta}_{i53}$, $\mathbf{\Theta}_{i34}$, 2000, 1500)

This record is converted into ($1+d$) cells as following:

3153**.**2000**.**1500

Where:

- 3, 1, 5, and 3 are the classes of the first, second, third, and fourth category attributes, respectively.
- 2000 and 1500 are the values of the first and the second data attributes, respectively.

C. Key Representation Query (KRQ)

Also, this scheme involves converting each new user query $\boldsymbol{q}$ from string representation into key representation query $\boldsymbol{q}^{\backslash}$, and stores it in the Audit Query table (AQ table). The key representation query $\boldsymbol{q}^{\backslash}$ contains, for each category attribute $\mathbf{A}_j$ ($1 \le j \le t$), either the specific category attribute class or the sign *. The * has the intuitive meaning 'any', namely all category attribute classes for the corresponding column. Also, the key representation query $\boldsymbol{q}^{\backslash}$ contains, for each data attribute $\mathbf{A}_j$ ($t+1 \le j \le t+d$), the value of the data attribute itself, or logical formula over its value or over any value with the same data type using the relational operators (>, ≥, <, ≤, =), or the sign *.

The key representation query $\boldsymbol{q}^{\backslash}$ will be as following:

$$\mathbf{u}_1\mathbf{u}_2\ldots\mathbf{u}_t\mathbf{.v}_1\mathbf{.v}_2\mathbf{.}\ldots\mathbf{.v}_d$$

Where:

- $\mathbf{u}_j$ = 1 | 2 | … | $|\mathbf{A}_j|$ | ***,*** ($1 \le j \le t$)
- $\mathbf{v}_j = \mathbf{V}_{ij}$ | **f(V)** | ***,*** ($1 \le i \le N$; $1 \le j \le d$), **f(V)** logical formula over any value **V** with the same data type.

21) The Relational Operators in KRQ: The relational operators in the key representation query q\ are signed as following:

TABLE I. THE RELATIONAL OPERATORS

Expression		**The sign in $q^{\backslash}$**
Greater than	>	→
Greater than or equal	≥	⟶
Less than	<	←
Less than or equal	≤	⟵
$\mathbf{v}_1 \le x \le \mathbf{v}_2$		$[\mathbf{v}_1,\mathbf{v}_2]$
$\mathbf{v}_1 < x \le \mathbf{v}_2$		$(\mathbf{v}_1,\mathbf{v}_2]$
$\mathbf{v}_1 \le x < \mathbf{v}_2$		$[\mathbf{v}_1,\mathbf{v}_2)$
$\mathbf{v}_1 < x < \mathbf{v}_2$		$(\mathbf{v}_1,\mathbf{v}_2)$

22) The Logical Operators in KRQ: The logical operators in the key representation query q\ are signed as following:

TABLE II. THE LOGICAL OPERATORS

Logical Operator	**The sign in $q^{\backslash}$**
and	•
or	+
not	——

23) Examples: Let t = 4 and d = 2, and consider the following user queries:

(1) q_1: $(A_2=\Theta_{24})\bullet(A_4=\Theta_{41})\bullet(A_5=2500)$

(2) q_2: $(A_1=\Theta_{12})\bullet(A_2=\Theta_{21})\bullet(A_6>3000)$

(3) q_3: $(A_2=\Theta_{21})\bullet(A_3=\Theta_{34})\bullet(2000<A_5<4000)$

(4) q_4: $\text{not}(A_3=\Theta_{32})\bullet\text{not}(A_4=\Theta_{43})\bullet(A_6\le 2000)$

(5) q_5: $(A_1=\Theta_{12})\bullet(A_2=\Theta_{21})+(A_1=\Theta_{13})\bullet\text{not}(A_2=\Theta_{23})$

The key representation queries (KRQs) for the above string representation queries will be as following:

(1) $q^{\backslash}_1$: *4*1.2500.*

(2) $q^{\backslash}_2$: 21.*.3000⟶**

(3) $q^{\backslash}_3$: *14*.(2000,4000).*

(4) $q^{\backslash}_4$: **23.*.2000⟵

(5) $q^{\backslash}_5$: 21.*.* + 33**.*.***

D. Audit Query Table (AQ table)

The Audit Query table (AQ table) is used for storing each

- 342 -

new legal key representation query (KRQ) q\. This table consists of the following columns:

- The key representation query: $\boldsymbol{q}^{\backslash}$
- The query result (aggregation result): $\mathbf{R}(\boldsymbol{q}^{\backslash})$
- The query-set size: $|\boldsymbol{q}^{\backslash}|$
- The latest query-set size: $\mathbf{L}|\boldsymbol{q}^{\backslash}|$

Since the SDB in this scheme is online and dynamic, the individuals' records of an SDB need to be inserted, deleted, and updated dynamically. Consequently, we use the latest query-set size column L|q\| to deal with the previously key representation queries q\s, which was legal. After updating the SDB, the snooper may repeat invoking one of the previously queries again. By using the value of this column we can decide whether or not this query could lead to the disclosure of the SDB.

IV. Example

A typical example of SDB can be illustrated based on the data held in Table III. In the SDB, the salary of specific individual should not be disclosed. Table III shows the original database D summarizing confidential information about employees. Each employee is classified in three categories and has one data attribute. The possible category attributes' values are as follows:

- Gender: {M, F} = {1, 2}
- Dept: {CS, EE, PE} = {1, 2, 3}
- Level: {BSc, MSc, PhD} = {1, 2, 3}

The possible data attribute's values are:

- Salary (in $): any integer ≥ 0

Table IV shows the key representation database (KRDB) $\boldsymbol{D}^{\backslash}$, which the conversion result of the original database $\boldsymbol{D}$ by converting the three category attributes (Gender, Dept, Level) into one cell ($\mathbf{U}_{i1}\mathbf{U}_{i2}\mathbf{U}_{i3}$), and the data attribute value $\mathbf{V}_{i1}$ will be separated by the sign '**.**'. The converted two cells are as following:

$$\mathbf{U}_{i1}\mathbf{U}_{i2}\mathbf{U}_{i3}.\mathbf{V}_{i1}$$

Where, the cell $\mathbf{U}_{i1}\mathbf{U}_{i2}\mathbf{U}_{i3}$ represents the category attributes' classes corresponding to the category attributes (Gender, Dept, Level). And $\mathbf{V}_{i1}$ represents the value of the data attribute (Salary).

TABLE III. The Original Database $\boldsymbol{D}$

RecNo	Name	Gender	Dept	Level	Salary
1	Adil	M	CS	MSc	200
2	Omer	M	EE	MSc	150
3	Sara	F	EE	MSc	250
4	Saria	F	CS	MSc	150
5	Samy	M	PE	MSc	180
6	Maisoon	F	PE	BSc	220
7	Gasim	M	CS	MSc	100
8	Ahmed	M	EE	MSc	180
9	Fatima	F	CS	PhD	30
10	Nasir	M	PE	BSc	200
11	Mahasin	F	EE	MSc	250
12	Khalid	M	CS	PhD	30

TABLE IV. The Key Representation Database (KRDB) $\boldsymbol{D}^{\backslash}$

RecNo	Record's key $\mathbf{U}_{i1}\mathbf{U}_{i2}\mathbf{U}_{i3}.\mathbf{V}_{i1}$
1	112**.**200
2	122**.**150
3	222**.**250
4	212**.**150
5	132**.**180
6	231**.**220
7	112**.**100
8	122**.**180
9	213**.**30
10	131**.**200
11	222**.**250
12	113**.**30

Examples of user queries $\boldsymbol{q}$s (using the statistical query ***sum***) for this database, which converted into key representation queries (KRQs) $\boldsymbol{q}^{\backslash}$s, are as follows:

TABLE V. Examples of User Queries Converted Into KRQs

User query $\boldsymbol{q}$	KRQ $\boldsymbol{q}^{\backslash}$	Query set	Answer	$\lvert\boldsymbol{q}^{\backslash}\rvert$
q_1 = M.CS	11*.*	{1, 7, 12}	330	3
q_2 = F.(CS+EE).MSc	212.* + 222.*	{3, 4, 11}	650	3
q_3 = M+$\overline{\text{CS}}$	1**.* + *$\bar{1}$*.*	{1, 2, 3, 5, 6, 7, 8, 10, 11, 12}	1760	10
q_4 = Salary>200	***.$\overrightarrow{200}$	{3, 6, 11}	720	3
q_5 = Salary≤150	***.$\overleftarrow{150}$	{2, 4, 7, 9, 12}	460	5
q_6 = F.CS.MSc	212.*	{4}	150	1
q_7 = $\overline{\text{F.CS.MSc}}$	$\overline{212}$.*	{1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12}	1790	11

V. Audit Stages

This paper provides presentation of an auditing method that can be used to repel the attacks of the snooper to the confidentiality of the individual data in the SDB. We propose three audit stages to protect online and dynamic SDBs from being disclosed. These stages enable the Key Representation Auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB, and a malicious user (snooper) cannot deduce confidential information about some individuals by stitching the answers of any legal queries.

Before implementing the audit stages, the new user query-set size $|\boldsymbol{q}|$ must fall in the allowable range [$\boldsymbol{n}$, $\boldsymbol{N}$-$\boldsymbol{n}$], for some positive integer $\boldsymbol{n}$. That is, the new user query $\boldsymbol{q}$ must satisfy the query-set-size control [1].

Query-Set-Size Control:

A user query $\boldsymbol{q}$ is permitted only if:

$$\boldsymbol{n} \le |\boldsymbol{q}| \le \boldsymbol{N}-\boldsymbol{n},$$

Where, $\boldsymbol{n}\ge 0$ is a parameter of a database.

Given a sequence of key representation queries $\boldsymbol{q}^{\backslash}_1, \boldsymbol{q}^{\backslash}_2, \ldots, \boldsymbol{q}^{\backslash}_h$ that have already been posed and stored in the Audit Query table (AQ table), and a new key representation query $\boldsymbol{q}^{\backslash}_{h+1}$.

$$\boldsymbol{q}^{\backslash}_i = \mathbf{U}_{i1}\mathbf{U}_{i2}\ldots\mathbf{U}_{it}.\mathbf{V}_{i1}.\mathbf{V}_{i2}.\ldots.\mathbf{V}_{id}, \ (1\le \boldsymbol{i}\le \boldsymbol{h})$$
$$\boldsymbol{q}^{\backslash}_{h+1} = \mathbf{U}_{h+1,1}\mathbf{U}_{h+1,2}\ldots\mathbf{U}_{h+1,t}.\mathbf{V}_{h+1,1}.\mathbf{V}_{h+1,2}.\ldots.\mathbf{V}_{h+1,d}$$

To decide whether the new key representation query $\boldsymbol{q}^{\backslash}_{h+1}$ should be answered or not, the following audit stages should be applied.

A. The First Stage

Consider a sequence of key representation queries q'_i ($1 \le i \le h$), and a new key representation query q'_{h+1}. For the sake of simplicity, we shall write X_{ij} ($1 \le i \le h$; $1 \le j \le t+d$) to denote both U_{ij} ($1 \le i \le h$; $1 \le j \le t$) and V_{ij} ($1 \le i \le h$; $1 \le j \le d$).

$$q'_i = X_{i1}X_{i2} \ldots X_{it}.X_{i,t+1}.X_{i,t+2}. \ldots .X_{i,t+d}, \ (1 \le i \le h)$$

$$q'_{h+1} = X_{h+1,1}X_{h+1,2} \ldots X_{h+1,t}.X_{h+1,t+1}.X_{h+1,t+2}. \ldots .X_{h+1,t+d}$$

We assume that $a = i$ and $b = h+1$, if $|q'_i| > |q'_{h+1}|$. Otherwise, $a = h+1$ and $b = i$.

Accordingly, the new key representation query q'_{h+1} should be prevented if the KRA found q'_i (for some $i \in \{1, 2, \ldots, h\}$) satisfies the following conditions:

The First Stage Conditions:

A new KRQ q'_{h+1} is prevented if:

(i) $|q'_a| - |q'_b| = 1$, and

(ii) Each X_{bj} ($1 \le j \le t+d$), in q'_b, corresponds, in q'_a, to either * or X_{aj} where $X_{aj} = X_{bj}$, namely each cell X_{aj} should not correspond to different value in q'_b cells.

If the KRA found q'_i (for some $i \in \{1, 2, \ldots, h\}$) satisfied the above conditions, by subtracting the cells of the query q'_b from its corresponding cells in q'_a, excluding the common cells between them, the result query q'_r would return one record.

The proposed First Stage Algorithm (FSA) is shown in Fig. 1.

Example 1: Assume that the following query has already been posed and stored in the AQ table:

q_1 = F

Then, q'_1 = 2**.* = {3, 4, 6, 9, 11}, $|q'_1| = 5$

And the new user query is posed as follows:

$q_2 = F.\overline{CS}.\overline{MSc}$

Then, q'_2 = 212.* = {3, 6, 9, 11}, $|q'_2| = 4$

By using the KRA, the new user query q'_2 should be prevented, since (i) $|q'_1| - |q'_2| = 1$, and (ii) each X_{2j} ($1 \le j \le 4$), in q'_2, corresponds, in q'_1, to either * or X_{1j} where $X_{2j} = X_{1j}$ ($1 \le j \le 4$).

By subtracting the cells of the query q'_2 from its corresponding cells in q'_1, excluding the common cells between them, the result query q'_r would return one record.

$q'_r = q'_1 - q'_2$ = 2**.* - 212.* = {4}

Example 2: Assume that the following query has already been posed and stored in the AQ table:

q_1 = PE

Then, q'_1 = *3*.* = {5, 6, 10}, $|q'_1| = 3$

And the new user query is posed as follows:

q_2 = BSc

Then, q'_2 = **1.* = {6, 10}, $|q'_2| = 2$

By using the KRA, the new user query q'_2 should be prevented, since (i) $|q'_1| - |q'_2| = 1$, and (ii) each X_{2j} ($1 \le j \le 4$), in q'_2, corresponds, in q'_1, to either * or X_{1j} where $X_{2j} = X_{1j}$ ($1 \le j \le 4$).

By subtracting the cells of the query q'_2 from its corresponding cells in q'_1, the result query q'_r would return one record.

$q'_r = q'_1 - q'_2$ = *3*.* - **1.* = *31.* = {5}

```
AQ = {q'_1, q'_2, ..., q'_h}
q'_i = X_i1 X_i2 ... X_it.X_i,t+1.X_i,t+2. ....X_i,t+d
Procedure First_Stage (q'_h+1)
Begin
    Prevent = False;
    For each q'_i in AQ
  Begin
    If (ABS(|q'_i|-|q'_h+1|) == 1)
        If (|q'_i|>|q'_h+1|)
            a = i;
            b = h+1;
        else
            a = h+1;
            b = i;
        endif;
        For each X_ij in q'_i
        Begin
            If (X_bj == X_aj)
                Prevent = True;
            elseif ((X_bj != '*') and (X_aj == '*'))
                Prevent = True;
            elseif ((X_bj == '*') and (X_aj != '*'))
                Prevent = True;
            else
                Prevent = False;
                break;
            endif;
        end;
        If (Prevent == True)
            Inform the SDB to prevent q'_h+1;
            return Prevent;
            endif;
        endif;
    end;
  If (Prevent == False)
    Inform the SDB to permit q'_h+1;
    return Prevent;
    endif;
End;
```

Figure (1): The First Stage Algorithm (FSA)

B. The Second Stage

If the new key representation query q'_{h+1} consists of p parts, for some positive integer p.

$$q'_{h+1} = q'_{h+1,1} + q'_{h+1,2} + \ldots + q'_{h+1,p}$$

For this stage we have the following two cases:

- ***Case 1:***

If one of the q'_{h+1} parts returns one record, namely $|q'_{h+1,k}| = 1$ (for some $k \in \{1, 2, \ldots, p\}$), in this case the new key representation query q'_{h+1} should be prevented.

That is because if the KRA permitted this query and the snooper poses another query q'_{h+2} with the same parts of q'_{h+1} excluding the kth part, then he can deduce the individual's information by subtracting the answers of the two queries.

- 344 -

But if at least two of its parts return one record for each part, namely $|q^{\setminus}_{h+1,k}| = 1$ and $|q^{\setminus}_{h+1,j}| = 1$ (for some $k, j \in \{1, 2,\ldots, p\}$ and $k \neq j$), in this case the new key representation query $q^{\setminus}_{h+1}$ should be permitted.

Example 3: The query q = F**.**CS**.**MSc uniquely identifies the employee Saria. A general tracker's query set size must fall in the range [$2n$, N-$2n$] [5][18], that is [4, 8] with n=2 and N=12. The formula T = M qualifies as a general tracker since | T | = 7. The snooper applies the following equation to discover the total sum of all salaries **S**.

S = **sum**(M; Salary) + **sum**($\overline{M}$; Salary)
= 1040 + 900 = 1940

A tracker is obtained by defining:

- A query:

q_1 = F**.**CS**.**MSc + M
Then, $q^{\setminus}_1$ = 212**.*** + 1****.*** = {4} + {1, 2, 5, 7, 8, 10, 12}
= {1, 2, 4, 5, 7, 8, 10, 12}
sum(q_1; Salary) = 1190

- And a second query:

q_2 = F**.**CS**.**MSc + $\overline{M}$
Then, $q^{\setminus}_2$ = 212**.*** + 1****.*** = {4} + {3, 4, 6, 9, 11}
= {3, 4, 6, 9, 11}
sum(q_2; Salary) = 900

The forbidden query q = F**.**CS**.**MSc can be computed using the following formula:

sum(q; Salary) = **sum**(q_1; Salary) + **sum**(q_2; Salary) – **S**
= 1190 + 900 – 1940 = 150
This is Saria's Salary.

By using the KRA, the query $q^{\setminus}_1$ = 212**.*** + 1****.*** should be prevented, since one of its parts $q^{\setminus}_{1,1}$ = 212**.*** returns one record. Also, the query $q^{\setminus}_2$ = 212**.*** + 1****.*** should be prevented, since one of its parts $q^{\setminus}_{2,1}$ = 212**.*** returns one record.

- ***Case 2:***

If one of the $q^{\setminus}_{h+1}$ parts, say $q^{\setminus}_{h+1,k}$ (for some $k \in \{1, 2,\ldots, p\}$), satisfied the *first stage conditions* with one of the KRQs, say $q^{\setminus}_i$ (for some $i \in \{1, 2,\ldots, h\}$), that has already been posed and stored in the AQ table. And if:

$$q^{\setminus}_{h+1,k} \cap q^{\setminus}_{h+1,j} = \Phi$$

($1 \leq j \leq p$ and $j \neq k$).

Then the new key representation query $q^{\setminus}_{h+1}$ should be prevented.

That is because if the KRA permitted this query and the snooper poses another query $q^{\setminus}_{h+2}$ with the same parts of $q^{\setminus}_{h+1}$ excluding the kth part, then he can deduce the individual's information by subtracting the answers of these queries.

```
q^\h+1 = {q^\h+1,1, q^\h+1,2, …, q^\h+1,p}
Procedure Second_Stage (q^\h+1, p)
Begin
  // Case 1 Second Stage
  counter = 0;
  Prevent = False;
  For each q^\h+1,k in q^\h+1
  Begin
    If (| q^\h+1,k| == 1)
      counter++;
    endif;
  end;
  If (counter == 1)
    Prevent = True;
    Inform the SDB to prevent q^\h+1;
    return Prevent;
  endif;
  // Case 2 Second Stage
  counter = 0;
  Prevent = False;
  For each q^\h+1,k in q^\h+1
  Begin
    Intersection = False;
    If (First_Stage (q^\h+1,k) == True)
      For each q^\h+1,j in q^\h+1
      Begin
        If (j != k)
          If (q^\h+1,k ∩ q^\h+1,j != Φ)
            Intersection = True;
            break;
          endif;
        endif;
      end;
      if (Intersection == False)
        counter++;
      endif;
    endif;
  end;
  if (counter == 1)
    Prevent = True;
    Inform the SDB to prevent q^\h+1;
    return Prevent;
  else
    Prevent = False;
    Inform the SDB to permit q^\h+1;
    return Prevent;
  endif;
End;
```

Figure (2): The Second Stage Algorithm (SSA)

Example 4: Assume that the following query has already been posed and stored in the AQ table:

q_1 = PE
Then, $q^{\setminus}_1$ = *3***.*** = {5, 6, 10}, $|q^{\setminus}_1| = 3$
sum(q_1; Salary) = 600

And the new user query is posed as follows:

q_2 = BSc + MSc
Then, $q^{\setminus}_2$ = **1**.*** + ****2**.***
= {6, 10} + {1, 2, 3, 4, 5, 7, 8, 11}
= {1, 2, 3, 4, 5, 6, 7, 8, 10, 11}, $|q^{\setminus}_2| = 10$
sum(q_2; Salary) = 1880

Then the snooper can poses the following query:

q_3 = MSc

Then, q'_3 = **2.*

= {1, 2, 3, 4, 5, 7, 8, 11}, $|q'_3| = 8$

sum(q_3; Salary) = 1460

The snooper applies the following formula to deduce Samy's salary:

Salary = **sum**(q_1;Salary) – (**sum**(q_2;Salary) – **sum**(q_3;Salary))

= 600 – (1880 – 1460) = 180 ■

By using the KRA, the query q'_2 = **1.* + **2.* should be prevented, since one of its parts $q_{2,1}$ = **1.* satisfied the *first stage conditions* with the query q'_1 = *3*.*, and $q'_{2,1} \cap q'_{2,2} = \Phi$.

The proposed Second Stage Algorithm (SSA) is shown in Fig. 2.

C. The Third Stage

Since the SDB in this scheme is online and dynamic, the individuals' records of an SDB need to be inserted, deleted, and updated dynamically. After updating the SDB, the snooper may repeat invoking one of the previously queries again. For this stage we have the following two cases:

- ***Case 1:***

If the new key representation query q'_{h+1} is equal to one of the previously key representation queries which has already been posed and stored in the AQ table, namely $q'_{h+1}=q'_i$ (for some $i \in \{1, 2,\ldots, h\}$). In this case the KRA will compare between the query-set size column $|q'_i|$ and the latest query-set size column $\mathbf{L}|q'_i|$.

The Third Stage Condition:

A new KRQ q'_{h+1} is permitted if:

$$(|q'_i| - \mathbf{L}|q'_i| = 0) \textbf{ or } (\mathrm{ABS}(|q'_i| - \mathbf{L}|q'_i|) \geq n),$$

(for some $i \in \{1, 2,\ldots, h\}$)

Where, $n \geq 0$ is a parameter of a database.

If the above condition is satisfied, then invoking the query q'_{h+1} again is permitted. But if this condition is not satisfied, the query q'_{h+1} is prevented.

Example 5: Let us suppose that the user posed the query:

q = PE

Then, q' = *3*.* = {5, 6,10}, $|q'| = 3$, $\mathbf{L}|q'| = 3$

After inserting a new record (13, Farid, M, PE, MSc, 250), the $\mathbf{L}|q'|$ value will be ***4***.

If the user posed the query q = PE again,

Then, q' = *3*.* = {5, 6,10, 13}, $|q'| = 3$, $\mathbf{L}|q'| = 4$

By using the KRA and with n=2, the query q' = *3*.* should be prevented, since $\mathrm{ABS}(|q'_i| - \mathbf{L}|q'_i|) = 1$ doesn't satisfy the *third stage condition*.

```
AQ = {q'1, q'2, …, q'h}
q'h+1 = {q'h+1,1, q'h+1,2, …, q'h+1,p}
Procedure Third_Stage_Condition (q'm)
Begin
  If ((| q'm|-L| q'm|)==0) or (ABS(| q'm|-L| q'm|) >= n))
    return True;
  else
    return False;
    endif;
End;
Procedure Third_Stage (q'h+1)
Begin
  // Case 1 Third Stage
    Permit = False;
    For each q'i in AQ
  Begin
    If (q'h+1 == q'i)
        If (Third_Stage_Condition (q'i) == True)
            Permit = True;
            Inform the SDB to permit q'h+1;
            return Permit;
        else
            Permit = False;
            Inform the SDB to prevent q'h+1;
            return Permit;
        endif;
    endif;
  end;
  // Case 2 Third Stage
  counter = 0;
  Permit = False;
  For each q'h+1,k in q'h+1
  Begin
    Intersection = False;
    If (Third_Stage(q'h+1,k) == True)
        For each q'h+1,j in q'h+1
      Begin
            If (j != k)
                If (q'h+1,k ∩ q'h+1,j != Φ)
                    Intersection = True;
                    break;
                endif;
            endif;
      end;
      if (Intersection == False)
            counter++;
      endif;
    endif;
    end;
    If (counter >= 1)
        Permit = True;
    Inform the SDB to permit q'h+1;
    return Permit;
  else
        Permit = False;
    Inform the SDB to prevent q'h+1;
    return Permit;
  endif;
End;
```

Figure (3): The Third Stage Algorithm (TSA)

- ***Case 2:***

- 346 -

If the new key representation query q'_{h+1} consists of p parts, for some positive integer p.

$$q'_{h+1} = q'_{h+1,1} + q'_{h+1,2} + \ldots + q'_{h+1,p}$$

If one of its parts, say $q'_{h+1,k}$ (for some $k \in \{1, 2,\ldots, p\}$), is equal to one of the previously KRQs which has already been posed and stored in the AQ table, namely $q'_{h+1,k} = q'_i$ (for some $i \in \{1, 2,\ldots, h\}$). And if:

$$q'_{h+1,k} \cap q'_{h+1,j} = \Phi \qquad (1 \le j \le p \text{ and } j \ne k).$$

In this case the KRA will check the *third stage condition*. If the condition is satisfied, then the new user query q'_{h+1} is permitted, otherwise the query q'_{h+1} is prevented.

That is because if the KRA permitted this query and the snooper poses another query q'_{h+2} with the same parts of q'_{h+1} excluding the kth part, then he can deduce the individual's information by subtracting the answers of these queries.

Example 6: Let us suppose that the user posed the query:

q_1 = PE
Then, q'_1 = *3*.* = {5, 6, 10}, $|q'_1|$=3, $\mathbf{L}|q'_1|$= 3
sum(q_1; Salary) = 600

After inserting a new record (13, Farid, M, PE, MSc, 250), the $\mathbf{L}|q'|$ value will be ***4***.

If the user posed the query:

q_2 = PhD + PE
Then, q'_2 = **3.* + *3*.*
= {9, 12} + {5, 6, 10, 13}
= {5, 6, 9, 10, 12, 13}
sum(q_2; Salary) = 910

Then the snooper can poses the following query:

q_3 = PhD
Then, q'_3 = **3.* = {9, 12}
sum(q_3; Salary) = 60

The snooper applies the following formula to deduce Farid's salary:

Salary = **sum**(q_2;Salary) – **sum**(q_3;Salary) – **sum**(q_1;Salary)
= 910 – 60 – 600 = 250 ■

By using the KRA and with n=2, the query q'_2 = **3.* + *3*.* should be prevented, since one of its parts $q'_{2,2}$ = *3*.* is equal to one of the previously KRQs and ABS($|q'_i|$ - $\mathbf{L}|q'_i|$) = 1 doesn't satisfy the *third stage condition*.

The proposed Third Stage Algorithm (TSA) is shown in Fig. 3.

VI. Cost Estimation

The records of a table must be allocated to disk blocks, because a block is the unit of a data transfer between disk and memory [19]. The key representation database (KRDB) D\ needs substantially fewer blocks than does the original database D. That is because each KRDB record is typically smaller in size than an original database record since it has only two attributes; consequently, more KRDB records than original database records can fit in one block.

Suppose that the block size is B bytes. For the original database D of fixed-length records of size R bytes, with $B \ge R$, we can fit $bfr = \lfloor B/R \rfloor$ records per block. The value bfr is called the blocking factor for the table [19][20]. R can be computed as follows:

$$R = \sum_{j=1}^{t+d} \text{Size}(A_j) \qquad \text{bytes}$$

The number of blocks b needed for the original database of N records is $b = \lceil N/bfr \rceil$ blocks.

For the KRDB D' of fixed-length records of size R' bytes, we can fit $bfr' = \lfloor B/R' \rfloor$ records per block. R' can be computed as follows:

$$R' = t + d + \sum_{j=t+1}^{t+d} \text{Size}(A_j) \qquad \text{bytes}$$

Because for each KRDB record we have only t bytes for the t category attributes, d bytes for the d data attributes' separators, and d data attributes (A_{t+1}, A_{t+2}, …, A_{t+d}). The number of blocks b' needed for the KRDB of N records is $b' = \lceil N/bfr' \rceil$ blocks.

Table VI shows the parameters of the cost estimation for the original database D and the key representation database D'.

TABLE VI. The Parameters of the Cost Estimation

	Original Database D	KRDB D'
Record Size (bytes)	$R = \sum_{j=1}^{t+d} Size(A_j)$	$R' = t + d + \sum_{j=t+1}^{t+d} Size(A_j)$
Blocking Factor (Record/Block)	$bfr = \lfloor B/R \rfloor$	$bfr' = \lfloor B/R' \rfloor$
Number of Blocks (Blocks)	$b = \lceil N/bfr \rceil$	$b' = \lceil N/bfr' \rceil$

The following two examples illustrate the saving in block accesses (CPU time) and storage space that are attainable when a KRDB is used (see Table VII).

Example 1: The fixed-length records of the original database in Table III have a record size R = 4 + 20 + 1 + 2 + 3 + 4 = 34 bytes. Suppose that this table contains N=30,000 records on a disk with block size B=1024 bytes. The blocking factor for the table would be $bfr = \lfloor B/R \rfloor = \lfloor 1024/34 \rfloor$ = 30 records per block.

- The number of blocks needed for this table is $b = \lceil N/bfr \rceil = \lceil 30{,}000/30 \rceil$ = 1000 blocks.
- A linear search on this table would need b=1000 block accesses.
- To perform a binary search on this table, in case it is ordered table, would need approximately $\lceil \log_2 b \rceil = \lceil \log_2 1000 \rceil$ = 10 block accesses.
- If a table has to be sorted, we would have to add the cost of the sort, which would need approximately

$\lceil b \log_2 b \rceil$ = $\lceil 1000 \log_2 1000 \rceil$ = 9966 block accesses.

TABLE VII. THE KRDB D' VS THE ORIGINAL DATABASE D

	Original Database D	KRDB D'	Reduction (%)	Improvement
Record Size (bytes)	R = 34	R' = 8		
Blocking Factor (Record/Block)	bfr = 30	bfr' = 128		
Number of Blocks (Blocks)	b = 1000	b' = 235	76.5 %	**Vast**
Linear Search (Block Accesses)	b = 1000	b' = 235	76.5 %	**Vast**
Binary Search (Block Accesses)	$\lceil \log_2 b \rceil$ = 10	$\lceil \log_2 b' \rceil$ = 8	20.0 %	**Slightly**
Sorting (Block Accesses)	$\lceil b \log_2 b \rceil$ = 9966	$\lceil b' \log_2 b' \rceil$ = 1851	81.4 %	**Vast**

Example 2: Consider the KRDB D' in Table IV, which its fixed-length records have a record size R' = 3 + 1 + 4 = 8 bytes. Suppose that this table also contains N=30,000 records on a disk with block size B=1024 bytes. The blocking factor for the table would be bfr' = $\lfloor B / R' \rfloor$ = $\lfloor 1024/8 \rfloor$ = 128 records per block.

- The number of blocks needed for this table is b' = $\lceil N / bfr' \rceil$ = $\lceil 30{,}000/128 \rceil$ = 235 blocks. A vast improvement over number of blocks needed for the original database D, which required 1000 blocks.
- A linear search on this table would need b' = 235 block accesses. A vast improvement over the 1000 block accesses needed for a linear search on the original database D (See Fig. 4).
- A binary search on this table, would need approximately $\lceil \log_2 b' \rceil$ = $\lceil \log_2 235 \rceil$ = 8 block accesses. Slightly improvement over 10 block accesses needed for a binary search on the original database D (See Fig. 5).
- The cost for sorting this table would need approximately $\lceil b' \log_2 b' \rceil$ = $\lceil 235 \log_2 235 \rceil$ = 1851 block accesses. A vast improvement over the 9966 block accesses needed for sorting the original database D (See Fig. 6).

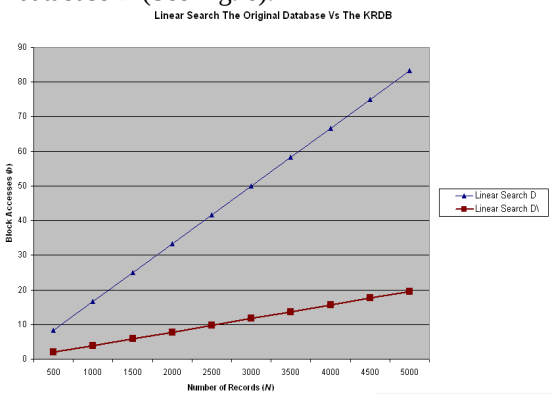

Figure (4): Linear Search: the Original Database Vs the KRDB

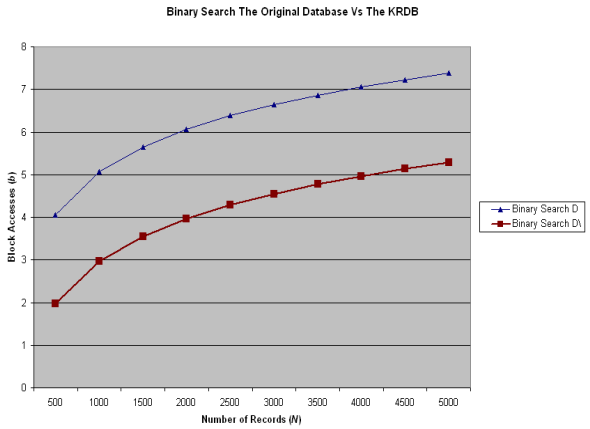

Figure (6): Sorting: the Original Database Vs the KRDB

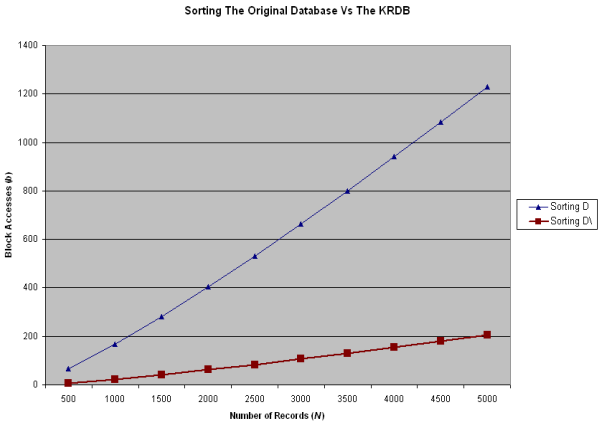

Figure (5): Binary Search: the Original Database Vs the KRDB

VII. CONCLUSION

The key representation auditing scheme is proposed to protect online and dynamic SDBs from being disclosed. The core idea of this scheme is to convert the original database, which in both string and numerical representations, into key representation database (KRDB). Also, this scheme involves converting each new user query from string representation into key representation query (KRQ), and stores it in the Audit Query table (AQ table). Three audit stages are proposed to protect the confidentiality of the individuals. In this paper, efficient algorithms for these stages are presented. These algorithms enable the Key Representation Auditor (KRA) to conveniently specify the illegal queries which could lead to disclosing the SDB. Also, cost estimation for this scheme is performed, and we illustrate the saving in block accesses (CPU time) and storage space that are attainable when a KRDB is used.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments.

REFERENCES

[1] S. Castano, M. Fugini, G. Martella, and P. Samarati, Database Security, 1st ed, Addison-Wesley, 1995, pp. 291-341.
[2] J. Wortmann, and N. Adam, "Security-Control Methods for Statistics Databases: A Comparative Study," ACM Computing Surveys, (Dec. 1989), Vol. 21(4) pp. 515-554.
[3] S. Shieh, and C. Lin, "Information Protection in Dynamic Statistical Databases," Invited paper for Encyclopedia of Computer Science and Technology, 1999.

- 348 -

[4] S. Shieh, and C. Lin, "Auditing User Queries in Dynamic Statistical Databases," Information Science, vol. 113(1-2), pp. 131-146, January 1999.
[5] D. Denning, Cryptography and Data Security, Addison-Wesley Publishing Company, Inc., 1982, pp. 313-387.
[6] E. Unger, S. McNulty,and P. Conell, "Natural Change in Dynamic Databases as a Deterrent to Compromise by Trackers," Digital Object Identifier 10.1109/CSAC.1990.143760, 1990, Page(s): 116-124.
[7] F. Malvestuto, M. Mezzini, and M. Moscarini, "Auditing Sum-Queries to Make a Statistical Database Secure," ACM Transactions on Information and System Security (TISSEC), Volume 9, Issue 1 (February 2006), Year of Publication: 2006, ISSN:1094-9224, Pages: 31-60.
[8] R. Ahlswede, and H. Aydinian, "On Security of Statistical Databases," Digital Object Identifier: 10.1109/ISIT.2006.261767, 2006, Page(s): 506-508.
[9] F. Chin and G. Ozsoyoglu, "Auditing and Inference Control in Statistical Databases," IEEE Trans. on Softw. Eng., (Apr. 1982), pp. 574-582.
[10] M. McLeish, "An Information Theoretic Approach to Statistical Databases and Their Security: A Preliminary Report," In Proceedings of the 2nd International Workshop on Statistical Database Management, 1983, pp. 355-359.
[11] J. Kam, and J. Ullman, "A Model of Statistical Databases and Their Security," ACM Trans. Database Syst. 2, 1, 1977, l-10.
[12] F. Malvestuto, and M. Moscarini, "Computational Issues Connected with the Protection of Sensitive Statistics by Auditing Sum-Queries," In Proc. Of IEEE Scientific and Statistical Database Management, 1998, pages 134-144.
[13] J. Kleinberg, C. Papadimitriou, and P. Raghavan, "Auditing Boolean Attributes," Journal of Computer and System Sciences, 2003, 6:244-253.
[14] K. Kenthapadi, N. Mishra, and K. Nissim, "Simulatable Auditing," In Proc. of ACM PODS, 2005, pages 118-127.
[15] S. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards Robustness in Query Auditing," Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), September, 2006.
[16] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kiernan, R. Rantzau, and R. Srikant, "Auditing Compliance with a Hippocratic Database," In Proc. of VLDB, 2004, pages 516-527.
[17] R. Huebner, "Automated Mechanisms for Controlling Inference in Database Systems", January, 2004.
[18] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed, Wiley, April 2008, pp. 172-179.
[19] R. Elmasri, and S. Navathe, Fundamentals of Database Management Systems, 5th ed, Addison-Wesley, 2007, pp. 463-591.
[20] T. Connolly, and C. Begg, Database Systems A practical Approach to Design, Implement, and Management, 3rd ed, Addison-Wesley, 2002, pp. 604-649.

Asim Abdallah Elshiekh received his B.Sc. in Statistics and Computer Sciences from Faculty of Mathematical Sciences, University of Khartoum, Khartoum, Sudan, in 1999, the M.Sc. in Computer Sciences from University of Khartoum, Khartoum, Sudan, in 2002. He is a lecturer in the Faculty of Mathematical Sciences, University of Khartoum since 2002. Presently he is doing his PhD in the area of Statistical Database Security at the Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Malaysia. His fields of interest are Statistical Database Security, Database Security, Data Storage and Query Processing, Distributed Databases, and Mobile Databases. He has published technical papers in International journals and conferences.

Dr. P.D.D.Dominic obtained his M.Sc degree in operations research in 1985, MBA from Regional Engineering College, Tiruchirappalli, India during 1991, Post Graduate Diploma in Operations Research in 2000 and completed his Ph.D during 2004 in the area of job shop scheduling at Alagappa University, Karaikudi, India. Since 1992 he has held the post of Lecturer in the Department of Management Studies, National Institute of Technology (Formally Regional Engineering College), Tiruchirappalli- 620 015, India. Presently he is working as a Senior Lecturer, in the Computer and Information Sciences Department, Universiti Teknologi PETRONAS, Malaysia. His fields of interest are Operations Management, KM, E-business and Decisions Support Systems. He has published technical papers in International, National journals and conferences.