(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 December 2005 (01.12.2005)

PCT

(10) International Publication Number
WO 2005/114478 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US2005/017456

(22) International Filing Date: 18 May 2005 (18.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/850,827 20 May 2004 (20.05.2004) US

(71) Applicant (*for all designated States except US*): **INGRIAN NETWORKS, INC.** [US/US]; 475 Broadway Street, Redwood City, CA 94063 (US).

(72) Inventor: **KOYFMAN, Andrew**; 1365 17th Avenue, San Francisco, CA 94122 (US).

(74) Agent: **COLEMAN, Brian R.**; Perkins Coie LLP, 101 Jefferson Drive, Menlo Park, CA 94025 (US).

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

(54) Title: ENCRYPTED TABLE INDEXES AND SEARCHING ENCRYPTED TABLES

(57) **Abstract:** The present invention teaches a variety of methods for building and searching secure, indexed database tables. Sensitive portions of the database tables and database indexes are encrypted, ordered and searched according to Boolean functions arranged to work with encrypted data. Also disclosed is a database management system that allows authorized users to build and search encrypted tables.

ENCRYPTED TABLE INDEXES AND SEARCHING ENCRYPTED TABLES

TECHNICAL FIELD

The invention relates to encrypted database search mechanisms. In particular, the present invention teaches building encrypted database indexes and fast searching of encrypted database tables using encrypted database indexes.

BACKGROUND OF THE INVENTION

Computer databases are a common mechanism for storing vast amounts of information on computer systems while providing easy access to users. A typical database is an organized collection of related information stored as "records" having "fields" of information. For example, a customer database may have a record for each customer. Each record contains fields designating specifics about the customer, such as first name, last name, date of birth, home address, credit information, credit card information, social security number, customer history, and the like. As can be seen, much of this information is of a sensitive nature and thus databases are often maintained in an encrypted form.

A database management system (DBMS) typically provides and manages access to the database (i.e., the data actually stored on a storage device) for the users of the system. In essence, the DBMS shields the database user from knowing or even caring about underlying details involved in management of the database. Typically, all requests from users for access to the data are processed by the DBMS. For example, records can be searched, information may be added or removed from data files, information retrieved from, or updated in, such files, and so forth, all without user knowledge of underlying system implementation.

In this manner, the DBMS provides users with a conceptual view of the database that is removed from the implementation level. The general construction and operation of a database management system is known in the art.

For enhancing the speed in which the DBMS searches, stores, retrieves, and presents particular data records, the DBMS usually maintains database indexes on one or more database tables. Database tables may include the complete contents of a database, but usually the DBMS creates one or more tables consisting only of those records that are most frequently accessed. Take for example FIG. 2 illustrating a table 30 having N records arranged in columns of fields which are row id 32, first name 34, last name 36, and date of birth (DOB) 38. One can readily appreciate that within a customer database, this type of information may be frequently accessed and searched.

Although the data in a database table may be of a sensitive nature, to enable fast searching such database tables are not typically encrypted. To search an encrypted table, the DBMS would first have to expend computing power decrypting the table, then perform the search and finally encrypt (again) the table. Hence, encrypting the database table prohibits fast searching as currently there is no mechanism for searching encrypted data.

A database index, typically maintained as a B-Tree (or B+Tree) data structure, allows the records of a non-sorted database table to be organized via nodes of the database index in many different ways, depending on a particular user's needs. A database index may be constructed as a single file storing ordered nodes of index key values together with unique index data. The index key values are a data quantity composed of one or more fields from a record which are used to arrange (logically) the database table records in some desired order (index expression). The index data are unique pointers or identifiers to the actual storage location of each record in the database table. Both are referred to internally by the system for locating and displaying records in a database table. Like the tables, for ease of searching and data management, current schemes require that the data within the database index be maintained in an unsecured transparent form.

FIG. 3 illustrates one type of b-tree index, a binary tree index 50, according the prior art. The binary tree index 50 includes a plurality of nodes 52 arranged in a hierarchical order. In the example of FIG. 3, each node 52 includes a transparent index key 54 and a transparent index data 56. The transparent index key 54 contains data that is used for hierarchical ordering of the index 50. In the case of the table 30, e.g., data from the DOB column 38 would be well suited for use as the transparent index key 54, the index 50 arranged according to age. The transparent index data 56 contains the payload data of the node 52. In the case of the table 30, e.g., data from the row id column 32 would be well suited for use as the contents of the transparent index data 56.

Searching for a particular record in a B-Tree index occurs by traversing a particular path in the tree. To find a record with a particular key value, one would maneuver through the tree comparing key values stored at each node visited with the key value sought. The results of each comparison operation, in conjunction with the pointers stored with each node, indicate which path to take through the tree to reach the record ultimately desired. Ultimately, a search will end at a particular leaf node which will, in turn, point to (i.e., will store a pointer to or identifier for) a particular data record for the key value sought. Alternatively, the leaf nodes may for "clustered indexes" store the actual data of the data records on the leaf nodes themselves.

An index allows a database server to find and retrieve specific rows much faster than it could without using the index. A sequential or linear scan from the beginning of a database table, comparing each record along the way, is exceedingly slow compared to using an index. There, all of the blocks of records would have to be visited until the record sought is finally located. For a table of even moderate size, such an approach yields unacceptable performance. As a result, virtually all modern-day relational database systems employ B-Tree indexes or a variant.

FIG. 1 illustrates a method 10 for generating and searching a table index utilizing transparent data according to the prior art. An initial step 12 populates a transparent database with a variety of data, including sensitive and possibly non-sensitive data. A next step 14 generates one or more transparent tables consisting of commonly searched data extracted from the transparent database. See FIG. 2 and the preceding description for more details on tables.

Continuing on with the method 10 of FIG. 1, a next step 14 generates a transparent index suitable for searching a transparent table. Once indexes have been formed, a step 18 provides for fast searching of the transparent tables utilizing the indexes formed. Then a step 20 encrypts the database for permanent storage.

The methods involving database tables and indexes, like that of FIG. 1, provide for fast table searching of data extracted from a database. Although indexes are widely used to improve DBMS performance, they expose sensitive data to attack. This is because searching methods of the prior art require that the database tables and database indexes be maintained in a transparent format in order to facilitate fast searching. Encrypting the indexes and tables would secure such data, but the computing power required to encrypt and decrypt these tables and indexes whole scale each time a search is required renders such a method unfeasible.

SUMMARY OF THE INVENTION

The present invention teaches a variety of methods for building and searching secure, indexed database tables. Sensitive portions of the database tables and database indexes are encrypted, ordered and searched according to Boolean functions arranged to work with encrypted data. Also disclosed is a database management system that allows authorized users to build and search encrypted tables.

BRIEF DESCRIPTION OF THE DRAWINGS

Prior Art FIG. 1 is a flow chart for a method for performing indexed based table searching.

Prior Art FIG. 2 shows a table of commonly searched data extracted from a database.

Prior Art FIG. 3 shows a b-tree for the table of FIG. 2.

FIG. 4 is a flow chart for a method of forming a searchable and encrypted database index in accordance with one aspect of the present invention.

FIG. 5 illustrates an encrypted table according to one embodiment of the present invention.

FIG. 6 illustrates an encrypted and ordered binary tree index in accordance with one embodiment of the present invention.

FIG. 7 is a block diagram of a computer system in accordance with yet another embodiment of the present invention.

DETAILED DESCRIPTION

FIGS. 1 – 3 show prior art.

FIG. 4 illustrates a method 100 for building a searchable encrypted index according to one aspect of the present invention. The database index may be a binary tree, a b-tree, or other suitable data structure. In one embodiment of the present invention, the database index is constructed as a single file storing index key values together with unique index data. The index key values are a data quantity composed of one or more fields from a record which are used to arrange (logically) the database table records in some desired order (index expression). The index data are unique pointers or identifiers to the actual storage location of each record in the database table.

An initial step 102 populates a database with transparent data. A next step 104 generates a transparent database table from data extracted from the database. Typically a set of database tables for different types of information and searching are formed. The transparent database may be like the database table 30 of Prior Art FIG. 2. The method 100 contemplates forming an index for one column of fields taken from the database table. However, those skilled in the art will readily recognize how the present invention can operate within more complicated structures.

A next step 106 populates a set of nodes with transparent data taken from the database table formed in step 104. In forming the set of nodes, the data contained in the indexed column (i.e., the column of fields that will be searchable by the index formed by method 100) serves as the index key values, and the row ID associated with each field is the index data. Each index key and index data form a node.

A step 108 encrypts each node using a first private key. The present invention contemplates encrypting at least the index key values of each node, although there is benefit in also encrypting the row ID values as will be described in the following paragraph. The index key values consist of sensitive information, and thus require encryption.

The row ID values, although not inherently sensitive information, provide an indirect route to sensitive information and are therefore subject to attack. By way of explanation, encrypting the row ID values reduces the amount of information that is transparent to parties doing a known plain text attack. This is particularly true when the indexed data has relevance to some particular transparent data found in the database table. For example, there is little benefit in an attacker determining that an undeterminable person has a particular birth date. However, knowing the name of the person associated with the birth date has value. An attacker that determines the values of some of the index keys in the database index knows the comparison for the remaining database keys. By way of example, if an attacker knows where in the database index lies the values of January 22, 1990 and January 24, 1990, then the attacker can determine which entries lie in-between, i.e., January 23, 1990. However, when the row ID values are encrypted, the attacker cannot associate the entries in between with the correct identity.

Once the nodes have been encrypted in step 108, a next step 110 orders the nodes. Ordering is generally done with a "less than" comparitor associated with the particular data type being ordered. For example, in ordering integers the less than comparitor is a simple Boolean '<' operation. The present invention contemplates generation of a "less than" comparitor function designed for use with encrypted data. One suitable pseudocode implementation follows:

```
Operator Less Than (A, B)
        A'=Decrypt(A);
        B'=Decrypt(B);
Perform Less Then Operation appropriate to A', B' data types.
```

Of course, other Boolean operators such as '>' can be used to arrange the nodes. In light of the present teaching, implementing such Boolean operators for use in the fast searching mechanisms of the present invention will be readily apparent to those skilled in the art.

FIG. 6 illustrates one example binary tree index 200 in accordance with one embodiment of the present invention. The database index 200 is in an ordered format and includes a plurality of nodes 202. However, the ordering of the nodes is meaningless unless one has access to the encryption functionality. Each node 202 includes an index key 204 and an index data 206. The embodiment of FIG. 6 contemplates encrypting both the index key 204 and the index data 206. Other embodiments of the present invention contemplate encrypting only the index key.

With further reference to FIG. 4, once the database index has been formed in encrypted and ordered format, like that shown in FIG. 6, a next step 110 encrypts the data fields of the indexed column of the transparent database table and forms a database table that is at least partially encrypted. FIG. 5 illustrates an example with a table 150 having a row ID column 152, a first name column 154, a last name column 156, and an encrypted DOB column 158. Preferably the database table fields are encrypted using a second private key, thus adding a layer of security so that encrypted information from the database index cannot be used to imply ordering on the database table.

Now that the database table and the database index are formed, a step 114 generates other encrypted database tables and database indexes as desired. As will be appreciated, the database may be fully represented in a plurality of indexed database tables that are encrypted as desired. The different tables are useful for fast searching on different types of data. The number of tables is application specific. A final step 116 then deletes the transparent database table and the transparent database. The DBMS may now utilize the encrypted index table for searching the encrypted table.

FIG. 7 illustrates a computer system 250 in accordance with yet another embodiment of the present invention. The computer system 250 includes a database management system (DBMS) 252, a cryptographic service engine 254, an encrypted database 256, an encrypted database index 258, an encrypted database table 260, and a database client 262. The DBMS 252 provides and manages access to the database 256 for the database client 262. All requests for access are processed by the DBMS 252. The cryptographic service engine 254 is used by the DBMS 252 to perform encryption and decryption functions. While shown as a distinct process, the cryptographic service engine 254 may well be a component of the DBMS 252.

The encrypted database index 258 and the encrypted database 260 are built and operate as described above with reference to FIGS. 4 – 6. As will be appreciated, the computer system 250 may be distributed across a network and a plurality of computers. Remote clients may access the DBMS 252 through the use of secure protocols such as SSL or TSL.

As performed in the prior art, searching for a particular record in a B-Tree index (or other tree like index) occurs by traversing a particular path in the tree. To find a record with a particular key value, one would maneuver through the tree comparing key values stored at each node visited with the key value sought. Unlike the prior art, the comparison functions here would involve decryption functions as described above.

Traversing the database index of the present invention would only be possible if the requesting user has authority to perform the necessary decryption functions. The results of each comparison operation, in conjunction with a decrypted form of the pointers stored with each node, indicate which path to take through the tree to reach the desired record(s). Ultimately, a search will end at a particular leaf node, which will, in turn, point to (i.e., store a pointer to or identifier for) a particular data record for the key value sought. Alternatively, the leaf nodes may for "clustered indexes" store the actual data of the data records on the leaf nodes themselves. In any event, the index and the tables in their encrypted format prevent the attacker from easily obtaining sensitive information.

CLAIMS

What is claimed is:

1. A computer implemented method for building an encrypted database index for searching a database table, said method comprising:

forming a database table including at least a row ID column having a plurality of row ID fields, and a sensitive data column having a plurality of sensitive data fields;

populating a plurality of nodes, each node having both an index key being data from a unique one of said plurality of sensitive data fields and an index data being data from a unique one of said plurality of row ID fields, whereby said nodes include a plurality of index keys and a plurality of index data;

encrypting said plurality of index keys found in said plurality of nodes; and

ordering said plurality of nodes.

2. A computer-implemented method as recited in claim 1, wherein said act of forming a database table includes forming a database table having a plurality of field columns.

3. A computer-implemented method as recited in claim 1, wherein said act of encrypting said plurality of index keys is performed using a first private key.

4. A computer-implemented method as recited in claim 3, further comprising encrypting said plurality of index data using said first private key.

5. A computer-implemented method as recited in claim 4, further comprising encrypting data from said plurality of row ID fields.

6. A computer-implemented method as recited in claim 5 wherein said act of encrypting data from said plurality of row ID fields uses a second private key.

7. A computer-implemented method as recited in claim 1 further comprising deleting the transparent data corresponding to said plurality of index keys so that only the encrypted plurality of index keys remains.

8. A computer-implemented method as recited in claim 1 further comprising storing said encrypted database table and said encrypted plurality of nodes to a persistent memory.

9. A computer readable medium comprising computer executable instructions for a "less than" comparitor function useful for comparing two encrypted data A and B, said computer executable instructions including:

receiving data A and B in an encrypted format;

decrypting A to form A';

decrypting B to form B';

performing a "less than" operation appropriate to a common data type of data A' and B'; and

returning results from said "less than" operation appropriate to a common data type of data A' and B'.

10. A computer readable medium comprising computer executable instructions for a "less than" comparitor function useful for comparing two encrypted data A and B as recited in claim 9, said computer executable instructions further including determining whether an entity requesting an operation of said "less than" function is authorized to perform encryption operations.

11. A computer readable medium comprising computer executable instructions for a "greater than" comparitor function useful for comparing two encrypted data A and B, said computer executable instructions including:

receiving data A and B in an encrypted format;

decrypting A to form A';

decrypting B to form B';

performing a "greater than" operation appropriate to a common data type of data A' and B'; and

returning results from said "greater than" operation appropriate to a common data type of data A' and B'.

2: A secure index for indexing a database table, said secure index searchable only by users authorized to utilize encryption functions available on said computer system, said secure index comprising:

a plurality of nodes, each node including an index key and an index data, each index key being encrypted, decrypted data from each index key providing meaningful information, and each index data identifying a storage location within said database table; and

wherein said plurality of nodes is logically ordered in a manner searchable only through decryption.

13. A secure index as recited in claim 12, wherein each of said index data are encrypted according to a first private encryption key.

14. A secure index as recited in claim 13, wherein data found in fields in said database table corresponding to said index data is encrypted according to a second private encryption key different from said first private encryption key.

15. A method for searching an encrypted database table via a secure index, said secure index having a plurality of nodes, each node including an index key and an index data, each index key being encrypted, decrypted data from each index key providing meaningful information, and each index data identifying a storage location within said database table, and wherein said plurality of nodes is logically ordered in a manner searchable only through decryption, said method comprising:

receiving a request to search said secure index from a user;

determining whether said user is authorized to utilize encryption functions;

when said user is authorized to utilize encryption functions, performing said requested search using one or more comparison functions operable to decrypt indexed and encrypted data.

16. A computer system comprising:

an encrypted database;

a database table including at least a row ID column having a plurality of row ID fields, and a sensitive data column having a plurality of sensitive data fields, said row ID column and said sensitive data column encrypted;

a database index corresponding to said database table, said database index including a plurality of nodes, each node having both an index key being data from a unique one of said plurality of sensitive data fields and an index data being data from a unique one of said plurality of row ID fields, whereby said nodes include a plurality of index keys and a plurality of index data, said plurality of index keys being encrypted, said plurality of nodes being ordered according to said encrypted index keys; and

a database management system instantiated on said computer system, said database management system operable to search said database table using said database index.

17. A computer system as recited in claim 16, wherein said computer system further comprises a cryptographic service engine operable to perform cryptographic functions as requested by said database management system.

18. A computer system as recited in claim 17, wherein said cryptographic service engine provides computer executable instructions for a "less than" comparitor function useful for comparing two encrypted data A and B, said computer executable instructions including:

receiving data A and B in an encrypted format;

decrypting A to form A';

decrypting B to form B';

performing a "less than" operation appropriate to a common data type of data A' and B'; and

returning results from said "less than" operation appropriate to a common data type of data A' and B'.

19. A computer system as recited in claim 17, further comprising an cryptographic API which provides an interface to said cryptographic service engine for applications running on said computer system.

20. A computer database having a database table and a database index, said computer database characterized in that: sensitive portions of said database table and said database index are encrypted, said database index ordered and searched according to Boolean functions arranged to work with encrypted data.

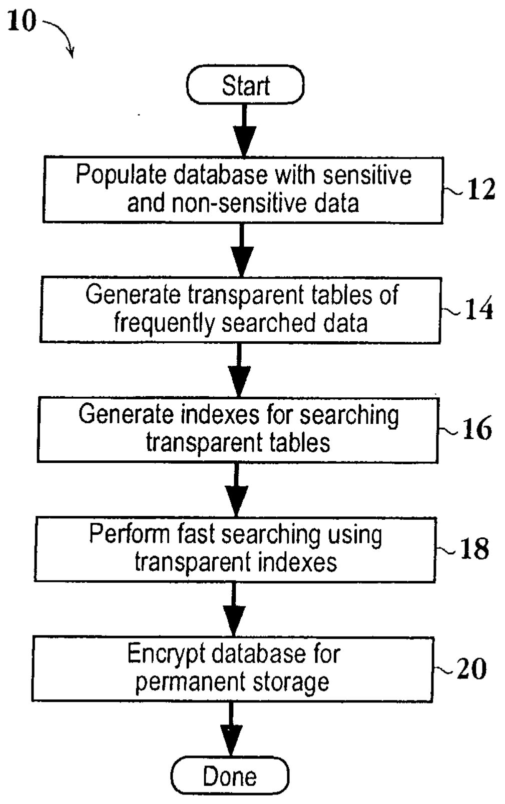

10
Start
Populate database with sensitive and non-sensitive data
12
Generate transparent tables of frequently searched data
14
Generate indexes for searching transparent tables
16
Perform fast searching using transparent indexes
18
Encrypt database for permanent storage
20
Done

Fig. 1 *(prior art)*

30

32	34	36	38
Row ID	*First Name*	*Last Name*	*DOB*
1	Andrew	Koyfman	3/15/85
2	Brian	Coleman	3/21/60
⋮	⋮	⋮	⋮
n	Sharyl	Brown	9/15/79

Fig. 2 *(prior art)*

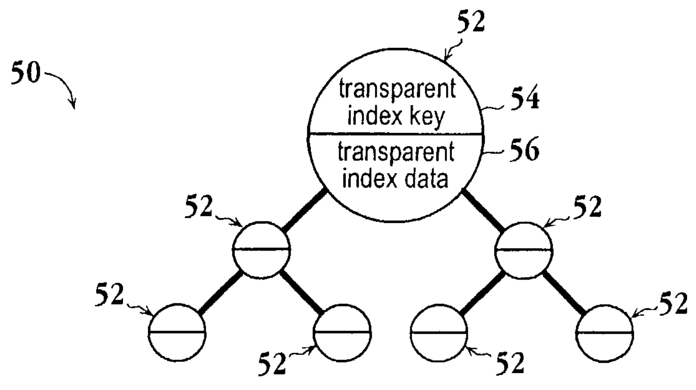


Fig. 3 *(prior art)*

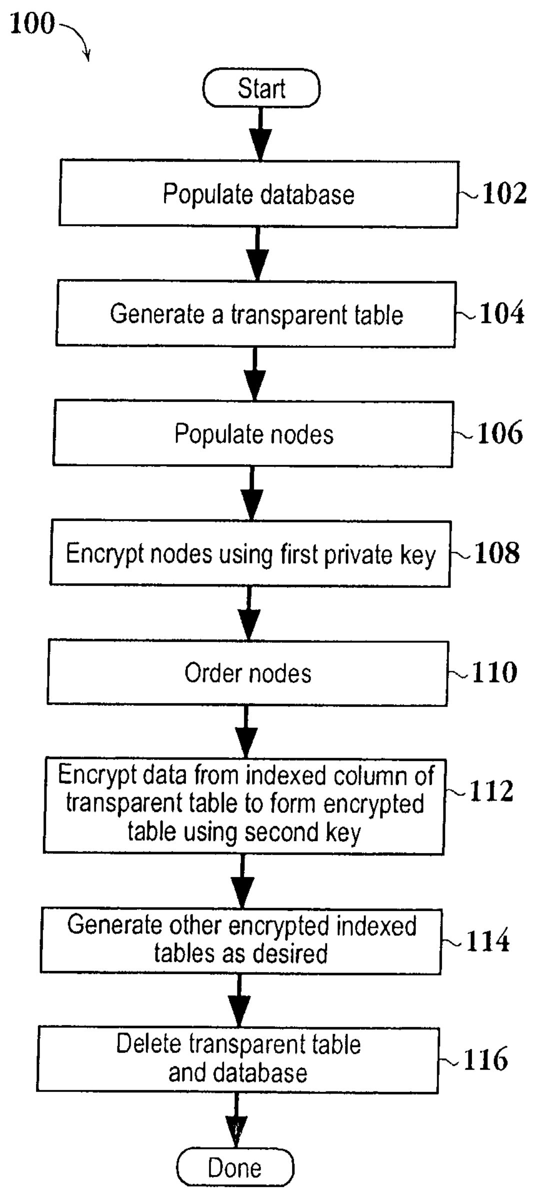
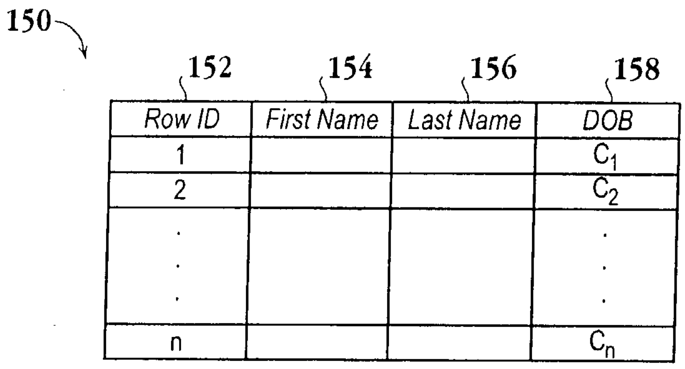

100
Start
Populate database
102
Generate a transparent table
104
Populate nodes
106
Encrypt nodes using first private key
108
Order nodes
110
Encrypt data from indexed column of transparent table to form encrypted table using second key
112
Generate other encrypted indexed tables as desired
114
Delete transparent table and database
116
Done

Fig. 4


150
152
154
156
158

Row ID	First Name	Last Name	DOB
1			C_1
2			C_2
. . .			. . .
n			C_n

Fig. 5

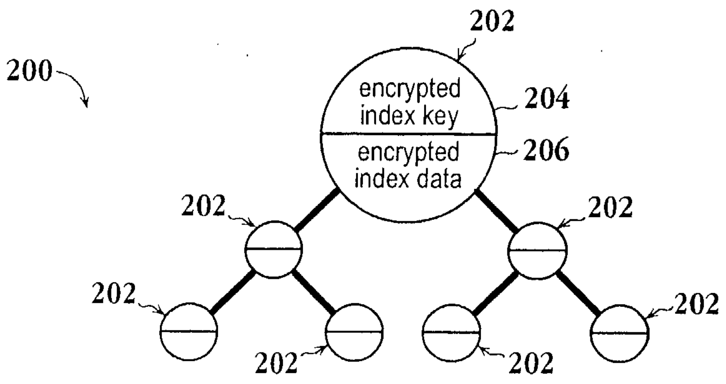


Fig. 6

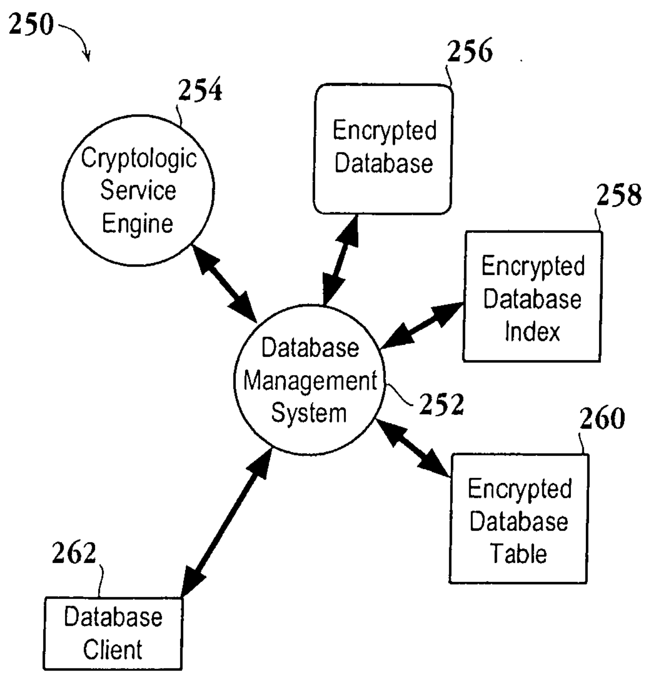

250
254
Cryptologic Service Engine
256
Encrypted Database
258
Encrypted Database Index
Database Management System
252
260
Encrypted Database Table
262
Database Client

Fig. 7