Database Steganography: Hiding Complete Database In Another Database.

Dr abdul Latif Ali Hussain
Iraqia University- Education College
Lateef1960@yahoo.com

ABSTRACT:
Data security is one of the most critical and an uttermost challenge in the digital world. Confidentiality, availability and integrity of data are demanded in every operation performed on data security. The data security in databases is also as important.
This paper has presents techniques for the data that can be implemented to immunize and reinforcement databases, and presents a new secure database system based on steganography that provides more confidentiality, authentication, and integrity during access of secrete data.
The proposed database system uses steganography technique to hide a database records inside another database records, and allows the authorized user to create tables and records which are hidden from unauthorized users.
The proposed system attempts to use the properties of the table file structure in a cover-database, and uses an embedded hidden database with hidden database management system to manage the hidden database. It can store and manage a large amount of data easily.

ملخص:
أمن البيانات أحد أهم التحديات واكثرها حرجا في العالم الرقمي. ثالوث المحدودية، الاتاحة، والتكامل للبيانات مطلوب في كل اجرائية تنفذ على أمن البيانات. أمن البيانات في قواعد البيانات مهم ايضا بنفس القدر. تقدم هذه الورقة البحثية تقنيات يمكن تطبيقها لتأمين وتحصين قواعد البيانات. وتقدم نظام قواعد بيانات آمن مؤسس على إخفاء البيانات يوفر محدودية، تخويل، وتكامل اكبر عند اجراء عملية الوصول للبيانات السرية.
نظام قواعد البيانات المقترح يستخدم تقنية إخفاء البيانات لإخفاء قيود قاعدة بيانات في قيود قاعدة بيانات أخرى، ويتيح للمستخدم المخول انشاء جداول وقيود مخفية عن المستخدمين غير المخولين. يروم النظام المقترح استخدام خصائص هيكل ملف الجدول في قاعدة البيانات- الغطاء، لاخفاء قاعدة بيانات مع نظام مخفي لادارة قاعدة البيانات المخفية. والهدف هو اتاحة اخفاء كمية كبيرة من البيانات وإدارتها.
أمن البيانات أحد أهم التحديات واكثرها حرجا في العالم الرقمي. ثالوث المحدودية، الاتاحة، والتكامل للبيانات مطلوب في كل اجرائية تنفذ على أمن البيانات. أمن البيانات في قواعد البيانات مهم ايضا بنفس القدر. تقدم هذه الورقة البحثية تقنيات يمكن تطبيقها لتأمين وتحصين قواعد البيانات. وتقدم نظام قواعد بيانات آمن مؤسس على إخفاء البيانات يوفر محدودية، تخويل، وتكامل اكبر عند اجراء عملية الوصول للبيانات السرية.
نظام قواعد البيانات المقترح يستخدم تقنية إخفاء البيانات لإخفاء قيود قاعدة بيانات في قيود قاعدة بيانات أخرى، ويتيح للمستخدم المخول انشاء جداول وقيود مخفية عن المستخدمين غير المخولين. يروم النظام المقترح استخدام خصائص هيكل ملف الجدول في قاعدة البيانات- الغطاء، لاخفاء قاعدة بيانات مع نظام مخفي لادارة قاعدة البيانات المخفية. والهدف هو اتاحة اخفاء كمية كبيرة من البيانات وإدارتها.

***KEYWORDS*:** Steganography, Database security, DBMS, Security techniques, Cryptography

1. Introduction

The database is a collection of related data organized in a way that can be easily stored, accessed, managed and updated, constructed from set of tables which are structured files. A database management system (**DBMS**) is a set of interdependent data and a set of programs to access those data [1]. A major objective of a database management system is to support users to get an abstract view of data, hiding specific details of how data is manipulated and stored, [2].

The data requires some security to preserve it where the level of security depends on the nature of these data. The security of data is very important issue for both organizations and individual purposes. The security issue must be characterized by ease of maintenance and having low overhead on system resources.

The database security is substantial because they encounter security threats that may prove disastrous if exposed or accessed publicly. Database system security is more than securing the database, and to achieve a secure database system, we need to secure: [3]

- Database tables.
- Database management system .
- Applications development.
- Operating system in relation with database system.
- Web server in relation with database system.
- Network environment in relation with database system**.**

1.1 Recent Used Database Security Techniques [4]:

- Cryptography: Cryptography is the techniques in which the plain text is converted to a non-readable and fog text by encryption.
- Hashing: Hashing is the transformation of a variable length data into a fixed length string using hash functions. The data retrieval is impossible without these hash functions.
- Access Control: Access control mechanisms prohibit the access to the database to outsiders except for the authorized users.
- Steganography: Steganography is the technique of concealing sensitive data or message in any type of cover media, so that the presence of data or the message itself is hidden. Figure (1).

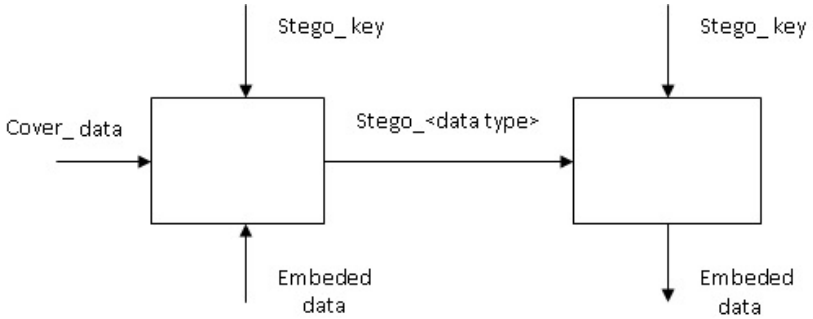


Figure (1)

2. Securing Database using Steganography

Related works

Steganography Premium: A software called (Steganography Premium), 2004, uses a file of the extension (dbf) as a cover-table. This attempt did not use the properties of the structure of the table file in the cover-table, but it is dealing with the table file as a plain text file. The software attaches the hidden information at the end of the table in a form of series of bytes, after it changes the size of the table to the new size. This method is very weak in maintaining against the removal of confidential information attack, because all confidential information will be deleted from the table file in the first use of the order (Delete / Back) when the user wants to delete any entry from the table. As well as a change of concealment process occurs in the table file size, which may raise attention if confidential data was large to some extent.[6].

[R. Rejani et al, 2013] The technique proposed in the paper creates a parallel data structure based on JSON to store information within a JPEG image, embedded in database field, using LSB based steganography for this purpose. While at the same time it is possible to view the image in a regular image viewer or any other software. [7].

[Radu Sion et al ,2004] In the paper, the authors introduce a solution for relational database content rights protection through watermarking numeric relational content. A significant point should be considered about watermarking. A watermark modifies the watermarked item. Therefore, a watermark cannot be embedded if the watermarked object cannot be modified then. Avoiding change the data is not the critical issue, but to limit the change to acceptable levels with respect to the intended use of the data.**[8].**

[Almusa, 2006]The aim was to represent a technique to hide sporadic data in a memo field in a record of database tables of types (dbf, dbc,lbx …etc) [9],[10].

3. Database Environment

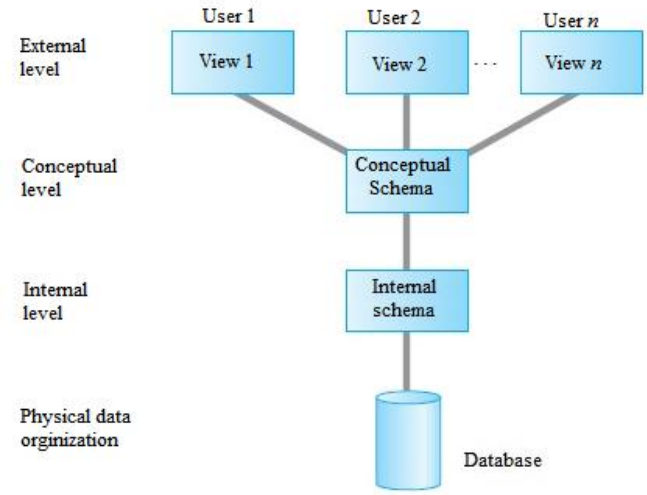
The database is a shared resource therefore each user may needs a different view of the data contained in the database. In general, to fulfill these needs, the architecture of commercial DBMSs available nowadays is based to some extent on the (American National Standards Institute, Standards Planning And Requirements Committee) ANSI-SPARC architecture. The ANSI-SPARC model did not become a standard; however it still provides a basis for explain some of the functionality of a DBMS.

The ANSI-SPARC model presents a **three-level architecture** containing an **external**, **a conceptual**, and an **internal** level, as illustrated in Figure (2).The **conceptual level** affords both the **mapping** and the demanded **independence** between the internal and external levels. The three-level architecture separates each user's view of the database from the way that is physically represented by the database.

Many reasons make this separation desirable:

- Users should be allowed to access the same data with a different customized view of that data.

- Users should be allowed to change the way they view the data, without affect each other.
- Users should be interacted with the database independently of storage considerations.
- Users should be prevented to deal directly with physical database storage details, such as hashing or indexing.
- The Database Administrator (DBA) should be allowed to change the storage structures of the database without altering the users' views.
- The changes to the physical aspects of storage should not be affecting the internal structure of the database, such as the changing to a new storage device.
- The DBA should be allowed to change the conceptual structure without affecting all users.



Fi

The users' view of the database is presented by the external level. This level identifies the part of the database which is relevant to each user. The in physical representation of the database on the computer is presented by the internal level. This level identifies the way the data is stored in the database. The internal level is involved with such issues as:

• Allocation of storage space for data and indexes;

• Descriptions of record for storage, and the stored sizes devoted to data items;

• Record positioning;

• Data encryption and data compression techniques.

A **physical level is** below the internal level that may be manipulated by the operating system under supervision of the DBMS.

The **database schema** is the general description of the database. Three types of schema in the database are defined according to the levels of abbreviation of the three-level architecture depicted in Figure (2), correspond to different views of the data there are multiple **external schemas**. At the conceptual level, there is the **conceptual schema**, which defines all the attributes, entities, and relationships simultaneously with integrity constraints. The **internal schema** is at the lowest level of abstraction, which is a complete definition of the internal model, containing the

definitions of methods of representation, the stored records, the data fields, and storage structures used and the indexes. Each database has only one internal schema and one conceptual schema. Mapping between the three types of schema is the responsibility of DBMS.

3.1 Components of DBMS

A DBMS is separated into many software components, each of which is appointed a specific operation. Figure (3) depict the main software components in a DBMS environment. The diagram illustrates the interface between the DBMS and the other software components, such as access methods and user queries (storing and retrieving data records as techniques for file management).

i. *Query processor:* A main DBMS component that assigned to transform queries into a series of low-level instructions conducted to the database manager

ii. *Database manager:* The *(DM)* interfaces with user-submitted queries and application programs. The DM receives and accepts queries and examines the conceptual and external schemas to ascertain required conceptual records that satisfied the request. The DM then sends a call to the file manager to execute the request.

iii. *File manager:* The file manager manages the underlying storage files and manipulates the allocation of storage space on disk. It creates and conserves the list of structures and indexes defined in the internal schema. It moves the requests to the proper access methods, to read data from or to write data into the system buffer.

iv. *DML preprocessor:* DML statements embedded in an application program is converted by this module into standard function calls in the host language. To generate the appropriate code, the DML preprocessor must interact with the query processor.

v. *DDL compiler:* DDL statements are converted by the DDL compiler into a set of tables containing metadata. These tables are stored later, in the system catalog. Control information is stored in data file headers.

vi. *Catalog manager:* This module manages access to and maintains the system catalog. Most DBMS components access the system catalog.

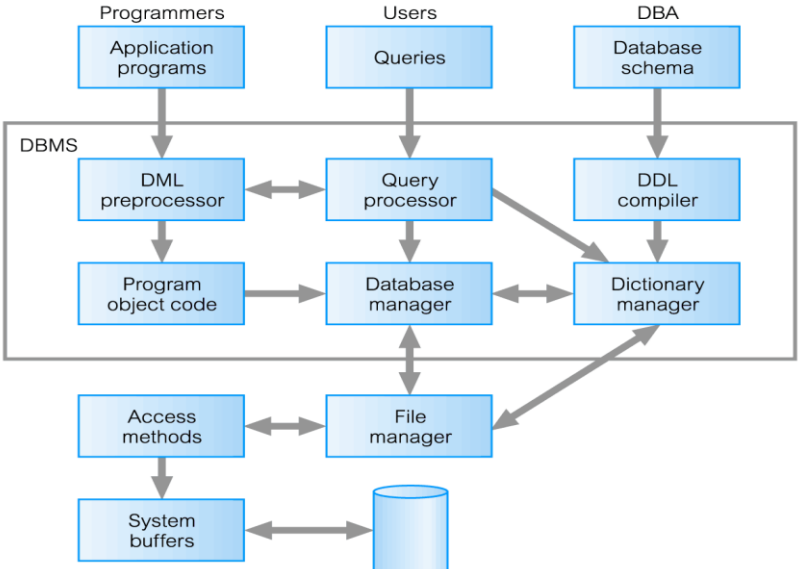


Figure (3) Components of DBMS [1]

Figure (3) Components of DBMS

3.2 dBASE file

The dBASE database management system used **DBF** file format to store data in tables and approved later by similar DBMS packages. The **DBF** format is not documented publicly and completely, and later versions of dBASE software extended the origin format to support new functionality [12] [13] [14].
The DBF file format developed by Ashton-Tate, but is understood by FoxPro,Act!, Clipper, Arago, xBase, Wordtech, and database-related or similar database products. Microsoft Excel and Microsoft Access can also open DBF files [14].

3.3 Binary, Memo, OLE Fields and .DBT Files

In DBF file, a separate .dbt file stored variable-length fields, such as binary and memo fields. The values in the .dbf file are index 10-byte entries into the .dbt file. Many applications of the DBF format did not support such data types in contexts unrelated to dBASE or other general-purpose database management systems. [12], [13].

OLE, memo, and Binary fields store data in .dbt files composed of blocks numbered sequentially (0, 1, 2, etc. The head record resides in the first (512) bytes in the file and starts at position (0) of the file. It consists of a definition of number of text characters and the type. Text blocks or the general data follows head record in the file [14].

4. Proposed Database Steganography System

The proposed system attempts to use the properties of the table file structure of the .DBF file.
Due to the results from the techniques used in [**9**],[**10**]. The proposed system aims to develop a technique to hide a complete database table in another database table. The main idea is to hide structured data, rather than sporadic general types of data, in the slack area of the memo field in a record of database file.
The proposed data hiding will be in two methods:

- Hidden-database table with the same structure of the cover table, but with different data values.
- Hidden-database table with a different structure table of the cover table, with different data values.

A hidden database management system (hidden-DBMS) will be designed to manage this hidden database tables.

Unauthorized user can access **only** the data in the cover-database table through the cover-DBMS, and **cannot** access the hidden data by cover-DBMS while the authorized user can access the hidden data using a hidden database management system, hidden-DBMS; the compound database management system will construct the stego-DBMS, as shown in Figure (4).

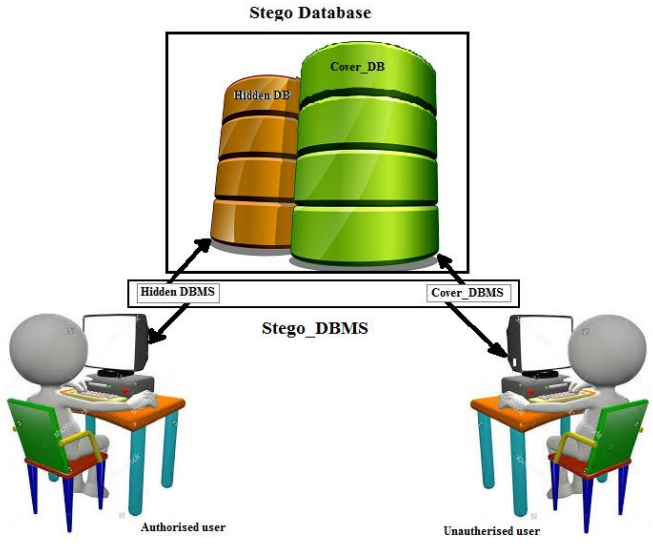
The proposed system will be constructed from stego-database, compound from cover-database and hidden-database. Each stego-database constructed from cover-tables and hidden-tables alternatively, these in turn, consist of stego-record constructed from cover-records and hidden -records alternatively.

4.1 Stego-Database Environment

For single database there is single conceptual schema and single internal schema. A stego-DBMS composite of two databases a cover-database and a stego-database. To satisfy the needs of a stego-DBMS with ANSI-SPARC model , there will be **three-level architecture** Consisting of a stego-**external**, **a stego- conceptual**, and a stego- **internal** level, as shown in Figure (5).

The **stego-conceptual level** provides the **independence** and **mapping** between the stego-external and stego-internal levels in the same manner of the original ANSI-SPARC model. The objective of the stego- architecture is expanded to separate the unauthorized user's view of the cover-database from the way the cover-database is physically represented, in the same time; it separates the authorized user's view of the hidden-database from the way the hidden-database is physically represented too.

This separation is desirable for the same reasons of original ANSI-SPARC model. For the proposed system purpose, this separation is more focused to give the DBA the ability to change the database storage structures without affecting the users' views, and the ability to change the conceptual structure of the database without affecting all users.



4.2 Components of Stego-DBMS

In a stego-DBMS, the cover-DBMS and the stego- DBMS are partitioned into the same software components depicted in figure (2) for the traditional DBMS, each of which has a specific operation. The main software components in a stego-DBMS environment are illustrated in Figure (6).This diagram shows how the cover-DBMS and stego-DBMS interface with other software components. The user queries in a stego-DBMS are of two types, authorized users queries and unauthorized users queries.

i. *Query processor:* Authorized users queries transforms queries into a series of low-level instructions conducted to the hidden-database manager. Unauthorized users queries transform queries into a series of low-level instructions conducted to the cover-database manager.

ii. *Database manager:* There will be two DMs. For cover-DBMS, the DM interfaces with user-submitted cover- application programs and unauthorized users queries. For hidden-DBMS, the DM interfaces with user-submitted hidden-application programs and authorized users queries. Each of the DMs accepts queries from its path, and inspects the external and conceptual schemas to ascertain required conceptual records that satisfied the request. The DM then sends a call to the file manager to execute the request.

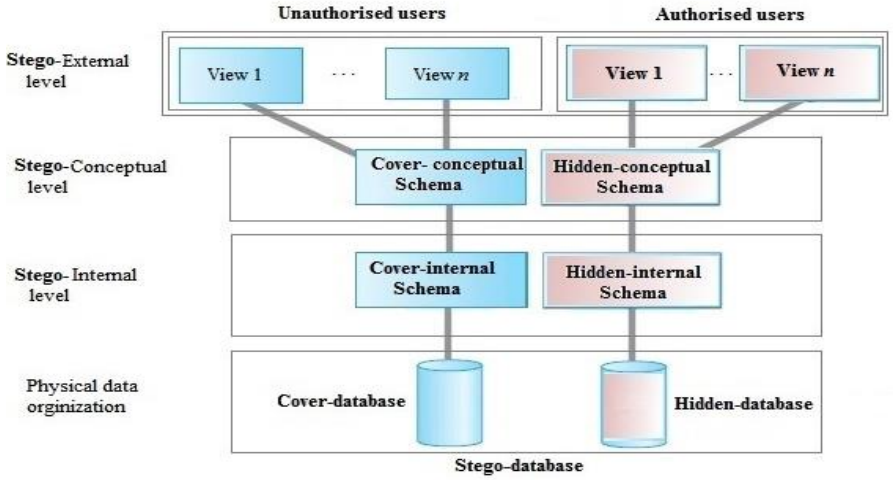


Figure (5) Stego-DBMS with ANSI-SPARC architecture

iii. *File manager:* The file manager manages the underlying storage files and manipulates the allocation of storage space on disk. It creates and conserves the list of structures and indexes defined in the stego-internal schema. It moves the requests to the proper access methods, to write data into or to read data from the system buffer.

iv. *DML preprocessor* Both of cover-DBMS and hidden-DBMS has its own DML preprocessor. The module in the cover-DBMS converts DML statements embedded in the cover-application program into standard function calls, while The module in

the stego-DBMS converts DML statements embedded in the hidden- application program into standard function calls. The DML preprocessor in the cover-DBMS must interact with the query processor of the cover-DBMS to generate the appropriate code and the DML preprocessor in the hidden-DBMS must interact with the query processor of the hidden-DBMS to generate the appropriate code .

v. *DDL compiler:* In both systems, the cover-DBMS and the hidden-DBMS, the DDL compiler converts DDL statements into a set of tables containing metadata. These tables are then stored in the system catalog of each system. Control information is stored in data file headers for the cover-DBMS. For the hidden-DBMS the control information must be managed and stored in the hidden file headers.

vi. *Catalog manager:* Each of the cover-DBMS and the hidden-DBMS has its own catalog manager. Each of them manages access to and maintains its system catalog.

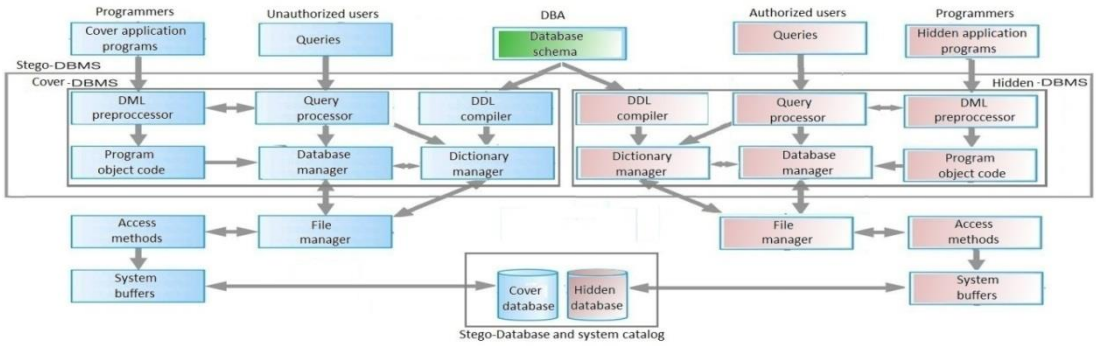


Figure (6) Components of Stego-DBMS

5. Illustration example for the proposed system

The following are illustration examples for the two cases of methods to hide and retrieve the data from the proposed system. The example consists of a (cover-table) structure with four fields for each record and the properties of the structure shown in the table below:

Table (1) Example of cover-table structure

Field name	Field	Field type
	length	
St_Name	30	Character
St_Date	10	Date
St_Value	3.2	Numeric
St_text	10	Memo

5.1 Hidden-database table with the same structure of the cover-table

When the hidden-database table has the same structure of the cover-table, each stego-record consists of a cover-record and a hidden-record. Both records have the same field's structure of the cover-table, except the memo field

in the cover-record. The hidden-record is embedded in the memo field of the cover-record, and represents substances for the same structure.

As a result of the proposed technique, there will be (46) bytes represent the complete hidden-record and they will be embedded in the hidden part of the memo field (St_text), of the cover-record. The first (30) bytes of the hidden data (hidden-record) will represent the first field (St_Name) value, the next (10) bytes will represent the second field (St_Date) value, and the next (6) bytes will represent the third field (St_Value) value.

The cover-record, which appeared to unauthorized user, will hold values differ from the hidden record. The block-Size unit, for the memo field, must be proper to the size of a single record defined in the table header.

In this case, and due to the similarity of the two records, the cover-record and the hidden-record, the role of the hidden-DBMS is limited to access the hidden data and retrieve/store it, in low-level manner. The other data processing will taken place by the cover-DBMS.

5.2 Hidden-database table with a different structure of the cover-table

When the hidden-database table has a different structure of the cover-table, the first hidden record (records), must hold the (file header) of the (hidden-table). In other word it contains the meta-data about record field's properties.

For the same example above, the first (hidden-record) will be the text:

M_ID 10.3 N, M_Date 10 D, M_price 5.2 N

and it represents the structure of the hidden table.

The rest of (hidden-records) will hold the substances of these fields for each record, in the same manner applied in the previous case. The deference is that the hidden-record structure may vary from the structure of the cover-record.

5.3 Stego-Database management system (Stego-DBMS)

The visual programming languages provide their control objects with visibility property, to hide an object at start up. Setting this property in code enables the programmer to hide, and later to redisplay, a control at run time in response to a particular event. This feature will be the corner stone, in the proposed (stego-DBMS) to build a hidden application programs in the hidden database management system (hidden-DBMS) interface, inside the cover application programs in the cover database management system (cover- DBMS) interface.

The cover-DBMS will manipulate the data stored in the cover-database in high-level access method, while the hidden-DBMS will manipulate the data in the hidden-database in low-level access method.

Database files used in the proposed system are variable structure file type. The file header is a system record occupies the first record in every variable structure file. The high-level access method uses the table header to determine and retrieves the substances contained in this table. Therefore cover-DBMS displays and accesses only a value of a field defined in the file header, as the smaller data unit in the database system. That is, cover-DBMS cannot access any data in the table, out of the structure defined in the file header. Thus the cover-

DBMS user, who is unauthorized user, cannot see and access the data stored outside the structure of the record.

The hidden-DBMS uses low-level commands and functions supported by the programming languages, which allows treating the table file as a stream of bytes, or even bits. For that, the hidden-DBMS can display and access any part of data contained in the table, even if it is out of the boundaries of the table structure, defined previously in the table header to describe records and fields of the table.

The hidden-DBMS can be triggered to operate, by indistinguishable hot spot (e.g. red object on red background with same gradient), its position is known only for authorized user. Hidden-DBMS constructed from a complete user interface set, completely or partially, differ from user interface set of the cover-DBMS.

The roll of hidden-DBMS is different for the two cases of hidden-database. In the first case, when the cover-database and the hidden-database have the same structure, hidden-DBMS reads the hidden-record in low-level and swap it with the cover-record then the cover-DBMS will treat it as an ordinary records.

In the second case, when the cover-database and the hidden-database have a different structures, hidden-DBMS reads the first hidden-record (records) in low-level , constructs a new table with a new file header, and creates a new empty records as needed. The next hidden-records will be moved to fill the empty records created earlier. The cover-DBMS may be not suitable to manipulate the new table because of the difference of the database structure between the cover-database and the hidden-database; therefore the hidden-DBMS must be designed to carry out this task.

6. Conclusion

6.1 Steganographic process evaluation

Three evaluation criteria are considered to measure the performance of a Steganographic system. First is the embedding efficiency of the stego-object (number of hidden bits embedded per embedding change), second is the embedding capacity of the cover-object, and the third is robustness [15].

6.2 Embedding efficiency of the stego-object

The most important feature in this method is that it does not change the size of the cover-record, and does not change the file size. Due to the difference in the area used to implement hiding operation; there will be no change in the cover-records size, resulted from hiding the hidden-records. A high embedding efficiency will be achieved.

6.3 Embedding capacity of the cover-object

The internal fragmentation means that the file uses more space than the stated size, because the amount of disks used should be multiples of the size of a specific record size. The internal fragmentation resulted from the difference between the nature of the data stored, on the one hand, and the nature of the file structure or the method used in the storage on the other hand. The ratio of the two sizes of these unused spaces, expresses the level of efficiency of the file structure and its suitability to the nature of the data or the stored information.

In the proposed information hiding system, the aim to achieve is high embedding capacity of the cover database (i.e. increasing the hiding ratio). Three criteria limit the embedding capacity of the cover-database:

1. Length of block size.
2. Length of stored data in the record.
3. Number of records.

Embedding capacity of the cover-database can be calculated by the equation

$$C = \sum_{i=1}^{n} (B - r_i)$$

Where

C = Embedding embedded capacity

B= Length of block size

r_i = Length of stored data in *i*th record

n = Number of records

A "bad" data analysis in database systems, serves the concealment process more than the efficient data analysis, because it would widely allows the existence of space which is not used in the records, that is a savings of largest space to hide secret information, and enhances the hiding ratio, while it considered as wasted in terms of storage efficiency of the data in database. Storage efficiency and hiding ratio are inversely proportional, figure (7).

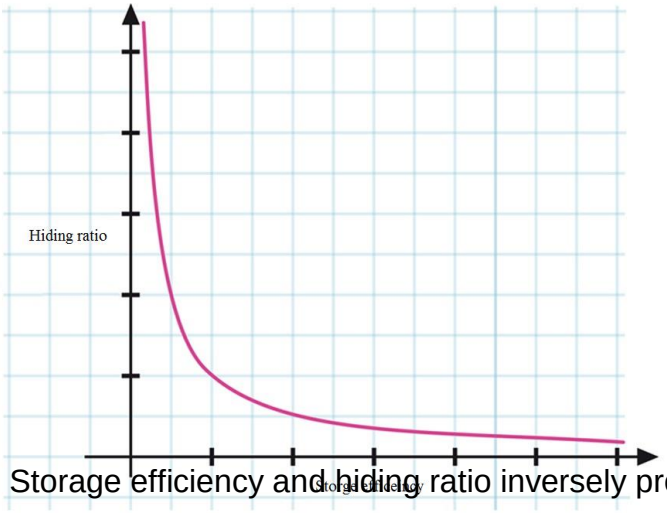


Figure (7): Storage efficiency and hiding ratio inversely proportional

From the perspective of the operating system, the dedicated storage space inside the file will no longer disk .

6.4 Stego-DBMS and stego-Database robustness

The robustness of the systems is high due to the inability to modify the hidden data out of the control of the hidden-DBMS.

The hidden data will not affected by any operation of the cover-DBMS. The exception is when the record is deleted entirely, then the data which is hidden in this record is deleted entirely too.

References

[1]. Abraham Silberschatz, Henry F. Korth, S.Sudarshan,"Database Systems Concepts", 5th Edition, 2006.

[2]. Thomas M. Connolly and Carolyn E. Begg "Database systems A Practical Approach

to Design, Implementation, and Management", 6th Edition, Pearson Education Limited 2005.

[3]. Paul J. Wagner "Database System Security ", University of Minnesota Summer School for Information Assurance, 2008.

[4]. Harshavardhan Kayarkar, Sugata Sanyal, "Classification Of Various Security Techniques In Database And Their Comparative Analysis", ACTA Technica Corviniensis, Vol. 5, Issue 2, April-June 2012, pp. 135-138.

[5]. Birgit P., "Information Hiding Terminology ", Information Hiding: First International Workshop, Proceeding, Springer, 1996

[6]. Steganography Premium, 2004. www.clickok.co.uk

[7]. R. Rejani, D. Murugan and Deepu V. Krishnan "Steganodb - A Secure Database Using Steganography", ICTACT Journal on Communication Technology, Septemper 2013, Vol: 04, Issue: 03.

[8]. Radu Sion, Mikhail Atallah and Sunil Prabhakar,"Rights Protection for Relational Data", IEEE Transactions on Knowledge And Data Engineering, Vol. 16, No. 6, June 2004.

[9]. Almusa A. A. Information hiding in database files using relative pointers],

[10]. Almusa A. A. "Information Hiding In Database Files and Operating Systems", A thesis Submitted to Al_Rasheed College of Engineering and Science, University of Technology April 2006.

[11]. Michael J. Folk, Bill Zoellick, Greg Riccadi, "File Structures: An Object-Oriented Approach with C++ 3rd Edition", Addison-Wesley Longman, Inc, 1998.

[12]. [https://www.loc.gov/preservation/digital/formats/fdd/fdd000325.shtml ,dBASE Table File Format (DBF)]

[13]. [http://www.dbase.com/KnowledgeBase/int/db7_file_fmt.htm Data File Header Structure for the dBASE Version 7 Table File]

[14]. [http://whatis.techtarget.com/fileformat/DBF-dBASE-file , DBF File Format]

[15]. R. Böhme, "Principles of Modern Steganography and Steganalysis", Springer, Volume 0 of the series (Information Security and Cryptography), pp17,2010.