Database Watermarking Using Bezier Curves

Major Project submitted in partial fulfillment of the
requirements for the award of degree of

Master of Technology

in

Information Systems

Submitted By:

Sakshi Goyal
(2K12/ISY/22)

Under the Guidance of:

Mr. Manoj Kumar
(Associate Professor)
(Department of Computer Engineering)

Department of Information Technology

Delhi Technological University

(2012-2014)

Certificate

This is to certify that **Sakshi Goyal (2K12/ISY/22)** has carried out the major project titled **"Database Watermarking Using Bezier Curves"** as a partial requirement for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2012-2014**. The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)
Mr. Manoj Kumar
Associate Professor
Department of Computer Engineering
Delhi Technological University
Bawana Road, Delhi-110042

Abstract

Digital Watermarking has been widely applied to relational database for ownership protection and information hiding. Digital watermarking for databases emerged as a candidate solution to provide copyright protection, tamper detection, traitor tracing, maintaining integrity of database. But robustness is a big challenge due to frequently database maintaining operators on those tuples. Here, we present an effective watermarking technique for relational database that is robust against various attacks. We present an approach which embeds a cubic curve into the database. To accommodate the cubic curve the original database is slightly modified by the embedding algorithm to obtain the watermarked database. The proposed system creates a cubic curve according to the parameters provided and then hides the cubic curve in the database by using an embedding algorithm. Embedding algorithm utilizes a parameter alpha and a key, which provides control for the hiding and recovery processes, restricting detection by those who do not possess the key, or do not have access to it.

Acknowledgement

I express my gratitude to my major project guide **Mr. Manoj Kumar, Associate Professor, Department of Computer Engineering** for the valuable support and guidance he provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members and my friends for providing their valuable help and time whenever it was required.

Sakshi Goyal
Roll No. 2K12/ISY/22
M.Tech (Information Systems)
E-mail: sakshigoyal09@gmail.com

Contents

List of Figures

List of Tables

Chapter 1: Introduction

Easy access to internet has boosted the growth of business and research. Nowadays, sharing information online is an important activity for business and research, which also involves buying/selling of databases. For example, sharing of data related to weather, stock market, power consumption, consumer behaviour, medical, scientific, etc. is frequently performed. Consequently, there is a great need for providing security of databases to discourage illegal copying and distribution in today's internet based application environment [1]. In this context, proof of ownership and tamper-proof-transportation of the databases are the most challenging issues these days [2].

Watermarking provides solutions for many of the problems encountered in the distribution of different multimedia objects such as, image, text, and audio [3]. Similarly, watermarking is effective in protection of relational databases [4]. Digital watermarking technique has been successively applied to protect the multimedia works and software products. Similarly, database watermarking has been proposed on large database security-control. However, there are some differences between relational database and multimedia. Firstly, a relational database table consists of many attributes and tuples, but there is no certain ordering between tuples or attributes of a relation table. Secondly, database maintaining operators could frequently change those tuples unlikely other type of multimedia object. Moreover, database tuples processing rely on logical set operational language such as SQL. So database watermarking should also have the ability of real-time update and blind detection and cannot directly adopt those multimedia watermarking method. It is more difficult to ensure the

robustness of database watermarking [5]. However, a common problem with relational database watermarking is the available bandwidth for watermark insertion, which is very limited compared to other multimedia objects. The data in relational databases can tolerate very small distortion. Increasing the amount of distortion may cause the values to become meaningless.

In recent years, scholars have carried out extensive research on database watermarking. The groundbreaking study in this area was conducted by R. Agrawal and R. Sion in 2002 [6, 7]. In 2003, X.M. Niu proposed that a meaningful string could be inserted into relational database as the watermark [8]. Y. J. Li raised a method of inserting watermark by changing the order of relational data index [9], and it does not change the physical location or data value to impair its use. Whereas the index is additional information outside of relational data content, watermark information could be completely lost if index of relational table is re-established or deleted. Y. Zhang converted image information into watermark cloud droplets according to D.Y. Li's cloud model idea, and then embedded it into relational data [10]. When being extracted, the cloud droplet should be compared with original copyright image. Moreover, Y. Zhang put forward a reversible watermarking method for relational database [11] that took the differences at the end of relational data and expanded it using wavelet transformation, then embedded watermark information. G. Gupta utilized difference expansion and Lowest-Effective-Bit on integers to achieve embedding and blind detection of watermark, but the method is only used for integer data that msake it not universal [12]. Many other watermark workers also make a lot of efforts to promote the development of database watermarking [13-24], yet there are still many shortcomings in current study. The watermark robustness is too weak to resist various conventional database operations and illegal watermark attacks, such as selection, addition, modification and so on. As a result, how to improve the robustness of database watermarking is a very difficult and significant work.

1.1 Motivation

The motivation for database watermarking is to protect databases, especially those published online (e.g., parametric specifications, surveys, and life sciences data), from tampering and pirated copies. A watermark can be considered to be some kind of information that is embedded into underlying data for tamper detection, localization, ownership proof, and/or traitor tracing purposes. Database watermarking techniques complement the Database Protection Act and are becoming increasingly important as people realize that "the law does not now provide sufficient protection to the comprehensive and commercially and publicly useful databases that are at the heart of the information economy".

1.2 Research Objective

The objective of this work is to present a technique for watermarking the relation databases that provides a high robustness that can resist various conventional database operations and illegal watermark attacks, such as deletion, addition, modification and so on. The algorithm will be explained in the forthcoming chapters with the experimental results.

1.3 Scope of work

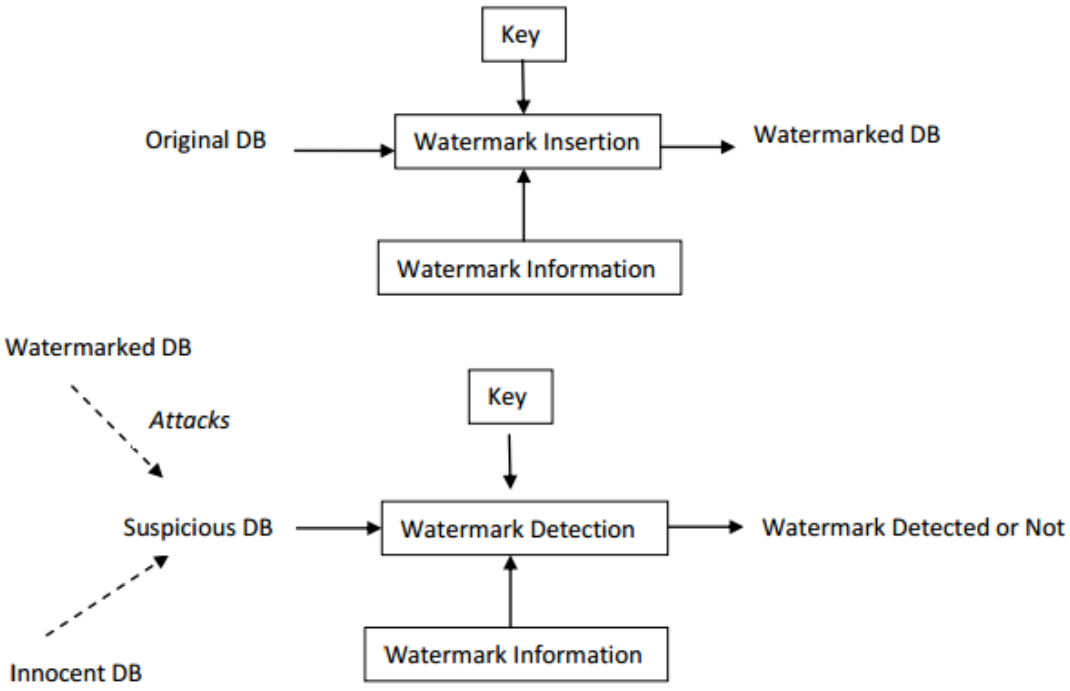
Here, the method of protecting relational databases copyright with Bezier curve watermark is studied only on the numerical attributes. In the future, we should expand our studies to non-numerical attributes fields. Because of the particularity of the relational data, there are some difficulties to research on watermarking relational databases in depth. But the right management of the relational databases copyright with watermark should be an important topic for database research.

1.4 Organization of thesis

In this chapter, I have highlighted the concept of database watermarking, motivation to do this thesis, my objective, and scope to do the work in same field. Chapter 2 provides a detailed picture of database watermarking and the prior work done till date. In chapter 3, I have presented the proposed scheme. Chapter 4 includes the implementation details and experimental results. Finally chapter 5 concludes the thesis.

Chapter 2:
Literature Review

Database watermarking consists of two basic processes: watermark insertion and watermark detection, as illustrated in Figure 2.1. For watermark insertion, a key is used to embed watermark information into an original database so as to produce the watermarked database for publication or distribution. Given appropriate key and watermark information, a watermark detection process can be applied to any suspicious database so as to determine whether or not a legitimate watermark can be detected. A suspicious database can be any watermarked database or innocent database, or a mixture of them under various database attacks.



(Figure 2.1: Basic Watermarking Process)

Based on whether the marking introduces any changes in the underlying data of the database, the watermarking techniques can be categorized into two [25]:

- Distortion-based
- Distortion-free

2.1 Distortion-based Watermarking

The watermarking techniques in this category introduce small changes in the underlying data of the database during embedding phase. The degree of changes should be such that any changes made in the data are tolerable and should not make the data useless. The watermarking can be performed at bit level, or character level, or higher such as attribute or tuple level, over the attribute values of types numeric, string, categorical, or any.

2.1.1 Watermarking Based on Numerical Data Type Attribute

(a) Arbitrary meaningless bit pattern as watermark information

The watermarking schemes proposed (also known as AHK algorithm) is based on numeric data type attribute and marking is done at bit-level [6, 26, 27]. The basic idea of these schemes is to ensure that some bit positions for some of the attributes of some of the tuples in the relation contain specific values. This bit pattern constitutes the *watermark*. The tuples, attributes within a tuple, bit positions in an attribute, and specific bit values at those positions are algorithmically determined under the control of the private parameters γ, ν, ξ and K known only to the owner of the relation. The parameters γ, ν, ξ and K represent number of tuples to mark, number of attributes available to mark, number of least significant bits available for marking in an attribute, and secret key respectively. The cryptographic MAC function $H(K \,||\, H(K \,||\, r.P))$ where $r.P$ is the primary key of the tuple r and || represents concatenation operation, is used to determine candidate bit positions. The HASH function H

(K || $r.P$) is used to determine the bit values to be embedded at those positions. The choice of MAC and HASH is due to the one-way functional characteristics and less collision probability. Pseudorandom sequence generator (*e.g.*, Linear Feedback Shift) is used instead of HASH and MAC to identify the marking bits and mark positions. The security and robustness of this scheme relies on these parameters which are completely private to the owner. The watermark detection algorithm is blind and probabilistic in nature. The relation is considered as pirated if the matching pattern is present in at least τ tuples, where τ depends on the actual number of tuples marked and a preselected value α, called the significance level of the test. Observe that the success of watermark detection phase depends on the fixed order of attributes. Re-sort of attributes' order may yield to the detection phase almost infeasible. Although the main assumption of this scheme is that the relation has primary key whose value does not change, they also suggest an alternative to treat a relation with- out primary key. Li et al. also suggest three different schemes to obtain virtual primary key for a relation without primary key.

Lafaye describes the security analysis for the AHK algorithm where he analyzes the security and robustness in two situations:

(i) Multiple Keys Single Database (MKSD): When a single database is watermarked several times using different secret keys and sold to different users, and

(ii) Single Key Multiple Databases (SKMD): When several different databases are watermarked using a single secret key. An attempt of random attack on a watermarked content obtained by the AHK algorithm, may be successful when randomize the ξ^{th} least significant bits of all tuples of the relation. However, this attack is highly invasive since most values of the relation are impacted by the attack. The locations guessed based on MKSD and SKMD can be used to build a better focused attack.

An improvement is suggested over the Agrawal and Kiernan's scheme [6]. Instead of using hash function, they use chaotic random series based on the Logistic chaos equation which has two properties: the non-repetitive iterative operation and the sensitiveness to initial value. It avoids the inherent weakness of collision of Hash function. The selection of bits of LSB for embedding watermark meets the requirements of both data range and data precision of each attribute, rather than simply to use a same ξ for all attributes. So the error caused by watermark is decreased significantly, hardly affects the availability of the database.

Among the most recent works, Gupta et al. [12] proposed a reversible watermarking scheme which is the modified version of Agrawal and Kiernan's one [6]. In this scheme, during the detection phase, the original unwatermarked version of the database can be recovered along with the ownership proof. The operation first extracts a bit *OldBit* from the integer portion of the attribute value before replacing it by the watermark bit and inserts it in the fraction portion of the attribute value. Thus, the watermark bit can be recovered during detection and the attribute can be restored to its unmarked value by replacing the watermark bit with the original bit *OldBit* extracted from the fraction part. They also propose another algorithm to defeat any attempt of additive or secondary attack which relies on the obvious fact that the database relation must be watermarked by the actual owner before Mallory.

Another watermarking method embeds random digits (between 0 to 9) at LSB positions of the candidate attributes for some algorithmically chosen tuples. During embedding phase, the tuples are securely partitioned into groups using the cryptographic hash function and only the first m (which is equal to the length of the owner's watermark) groups are considered. The decision whether to mark i^{th} ($1 \leq i \leq m$) group depends on the i^{th} bit of the owner's watermark, whereas the selection of the tuples in a group is based on a secret key (which is different from that used during partitioning) as well as the information at second LSB positions of the

numeric candidate attributes. Finally, for the selected tuples random numbers (between 0 and 9) are embedded at LSB positions in the attribute values of those tuples. Observe that although the owner has a watermark of length m, it is not actually embedded. Rather, it is used to identify some valid groups to embed the random values which acts as embedded watermark information. The detection phase determines the presence of mark in a group if the maximum occurrence frequency for a value between 0 and 9 for that group exceeds a threshold.

(b) Image as watermark information

Here, instead of embedding original image as watermark, an scrambled image based on Arnold transform with scrambling number d is used. Since Arnold transform of an image has the periodicity P , the result which is obtained in the extraction phase can be recovered from the scrambled form to the original after $(P - d)$ iterations. In the embedding phase, the original image of size $N \times N$ is first converted into scrambled image which is then represented by a binary string b_s of length $L = N X N$. Secondly, all tuples in the relation are grouped into L groups. The hash value which is computed using tuple's primary key, secret key and order of the image, determines the group in which each tuple belongs. Finally, the i^{th} bit of b_s is embedded into the algorithmically chosen bit position of the attribute value for those tuples in i^{th} group that satisfy a particular criteria. The detection phase follows majority voting technique. However, the security of this scheme improves as it relies not only on the secret key but also the scrambling number d and the order of the image N .

Rather than embedding scrambled image, another watermarking technique embeds the original image by first converting it into a bit flow (EMC, Encrypted Mark Code) of certain length, and then by following similar algorithmic steps. The only two differences are that (i) the first watermark insertion technique in assumes single fixed attribute to mark for all tuples

whereas the second one does not, and (*ii*) during selection of bit positions, the order of the image is not considered in the second scheme. Finally, after marking, it checks the usability of the data with respect to the intended use. If acceptable, the change is committed, otherwise rolled back.

Another watermarking scheme to embed image in BMP format is also presented. In watermark insertion phase, the BMP image is divided into two parts: *header* and *image data*. An error correction approach of BCH (Bose-Chaudhuri-Hocquenhem) coding is used to encode the image data part into watermark. Based on the tuple's ID value which is computed by performing hashing function parameterized with tuple's primary key and BMP header, all the tuples are assigned to k distinct subsets, where k is the length of the water- mark. Finally, each of the k bits of the watermark is used to mark each of the k subsets of tuples. During the marking, the selected least significant bit positions of selected attributes of some specific tuples satisfying a particular criteria are altered. The selection of bit positions depends on HASH or MAC function parameterized with BMP header, tuple's primary key and other parameters like number of least significant bits in the attribute value. Observe that the selected bit positions are not set to the watermark bits directly but rather, are set to mask bits which are computed from both the hash value and the watermark bit together.

The image-based fragile watermarking scheme aims at maintaining integrity of the database and uses support vector regression (SVR) to train high correlation attributes to generate the SVR predicting function for embedding watermark into particular numeric attributes. This scheme consists of three phases:

(i) Training Phase: Select training tuples and obtain trained SVR predicting function

(ii) Embedding Phase: All tuples in the relation are used to embed image watermark where the number of watermark bits is designed to be equal to the number of tuples.

Each numeric attribute value C_i of i^{th} tuple t_i is predicted using the SVR prediction function f resulting $\overline{C_i} = f(t_i)$. Based on the i_{th} watermark bit b_i (obtained after converting the image into bit flow), the value of C_i is modified by $\overline{C_i}$ +1 or $\overline{C_i}$ −1.

(iii) Tamper Detection: The trained SVR predicting function is used to generate the predicted value for each tuple and compared with the value contained in the database. The difference of these two values determines the watermark information and can ensure whether database is tampered or not. However, the limitation of this scheme is that it can identify the modification which takes place in the objective attribute set only. This scheme works good in the case where the tuples in the table are independent but highly correlated between the attributes.

(c) Speech as watermark information

Another scheme used the owner's speech to generate unique watermark. The preparation of watermark from the speech consists of several stages: compression of speech signal to shorten the watermark, speech signal enhancement to remove noise in frequency domain, speech signal conversion into bit stream, and finally, watermark generation by using the message of the copyright of the holder and the result of the converted speech signal. The bit-level marking is performed during watermark embedding phase by following the same algorithmic steps as in image-based technique.

(d) Genetic Algorithm based watermark signal

Genetic Algorithm-based technique is used to generate watermark signal, focusing on the optimization issue. They follow the same algorithmic framework as of image-based technique.

(e) Content characteristics as watermark information

Here, the watermarking schemes are performed based on the content of the database itself. The watermark insertion phase extracts some bits, called local characteristic, from the characteristic attribute A_1 of tuple t and embeds those bits into the watermark attribute A_2 of the same tuple. The selection of tuples depends on whether the generated random value (between 0 and 1) is less than the embedded proportion α of the relational databases and the non-NULL requirement of characteristic attribute value. In the watermark detection phase, by following similar procedure, the local characteristic of the characteristic attribute are extracted and compared against the last bits of watermark attribute.

A fragile watermarking scheme was proposed that can verify the integrity of database relation. In the proposed scheme, all tuples in a database relation are first securely divided into groups and sorted. In each group, there are two kinds of watermarks to be embedded: attribute watermark W_1 which consists of γ watermarks of length v and tuple watermark W_2 which consists of v watermarks of length γ, where γ and v are the number of attributes in a tuple and average number of tuples in each group respectively. W_1 and W_2 are created by extracting bit sequence from the hash value. For attribute watermark W_1, the hash value is generated according to the message authentication code and the same attribute of all tuples in the same group, while for tuple watermark W_2, it is formed from the same message authentication code and all attributes of the same tuple. Observe that, in both the embedding and detection phases, they ignore the least two significant bits of all attributes of numeric type except the primary key when computing hash values. The attribute watermark is embedded in LSB level, whereas tuple watermark is embedded at next to the LSB level. In this way, the embedded watermarks actually form a watermark grid, which helps to detect, localize and characterize modifications.

(f) Cloud model as watermark information

The cloud watermarking scheme is based on the Cloud Model with three digital characteristics: Expected value (*Ex*), Entropy (*En*) and Hyper Entropy (*He*). In watermark creation phase, it uses the forward cloud generation algorithm to generate cloud drops from cloud and embed those cloud drops into the relation as watermark, whereas the detection algorithm uses backward cloud generation algorithm to extract the cloud with parameters *Ex*, *En* and *He* from the embedded cloud drops, and finally, a similar cloud algorithm is used to verify whether both clouds (one used during watermark embedding and other extracted during watermark verification phase) are similar or not. This scheme is not blind as it requires original relational database during verification phase.

(g) Other meaningful watermark information

Another watermarking scheme embeds a meaningful watermark information by first converting it into a bit flow. The scheme computes unique ID for all tuples in the relation and sort them in ascending order according to their ID values. The tuples are then partitioned into p groups each containing m tuples. The i_{th} bit of the bit flow is embedded into the selected tuples in i_{th} group by following same selection criteria as in AHK algorithm with exception that it considers only single attribute to mark. Before committing the change, a constraint function is used to check whether the change exceeds the data usability bounds. The constraint function includes the basic data statistical measurement constraints, semantics constraints and structural constraints. Mean and standard deviation of the data set are very common aspects in basic data statistical measurement constraints. Semantics constraints and structural constraints are defined by user's input as SQL statements according to relational table. If during embedding phase, any tuple is selected but rolled back, it is recorded and

avoided during extraction phase to reduce false positive. Watermark extraction phase is blind, probabilistic and follows the majority voting technique.

The partitioning of tuples in most of the techniques is based on hashing. Another proposed scheme uses well-known techniques (*e.g. k-means* algorithm) to cluster the tuples into some equivalent classes. The embedding of the watermark bit is based on the comparison of the parity of watermark bit and the LSB of candidate attribute. The *k-means* method assures the location of the embedded watermark irregular.

The watermark insertion phase works in three phases:

(i) Select a group of candidates in all attributes of the relation, and record it as the watermark schema.

(ii) Append the error correction code (ECC Code) to the watermark.

(iii) Executes the watermark insertion algorithm. The insertion algorithm creates pseudo random sequence using primary key and secret key.

This sequence is used to identify the attribute to mark based on the significance of the attributes and the watermark bit to be embedded. During marking, the local constraint and a bidirectional mapping (to reduce watermarking data of various types into numeric data) are used. The local constraints can be defined as the upper bounds of "the distance" of attributes after/before watermarking.

Finally global constraints which is a series of SQL statements are evaluated to decide whether to commit the changes. Observe that watermark schema selection in embedding phase and watermark schema detection in verification phase exploit the non-blindness property.

A watermarking scheme follows same algorithmic steps as of image-based watermarking but embeds other meaningful watermark information rather than image by first converting it into

a bit flow of certain length. The selection of the candidate attribute can be based on the weights of all numeric attributes with a different hashing function. Observe that in watermark insertion phase the mark position is determined using the mark bit itself.

2.1.2 Watermarking Based on Categorical Data Type Attribute

Unlike the aforementioned watermarking schemes where the marking is based on numeric attribute, the right protection can be is based on categorical type data [2]. The watermark embedding process starts with a relation with at least a categorical type attribute A (to be watermarked), a watermark wm and a set of secret keys ($k1$, $k2$), and other parameters (e.g., e which determines the percentage of tuples to mark). Using the primary key K and secret key $k1$ and parameter e, it discovers a set of "fit" tuples, used to encode the mark. The fit tuple selection process is same as AHK algorithm. Suppose the database relation has η tuples, then fit tuples set contains roughly η/e tuples. The shorter watermark wm is converted into wm_data of length equal to η/e by deploying Error Correcting Code (ECC). The marking algorithm generates a secret value of required number of bits to represents all possible categorical values for attribute A depending on the primary key and $k1$, and then, forcing its least significant bit to a value according to a corresponding (random, depending on the primary key and $k2$) position in wm_data data. The pseudorandom nature of hash function H ($Ti(K)$, $k2$)) guarantees, on average, that a large majority of the bits in wm_data data are going to be embedded at least once. The use of different key $k1$ and $k2$ ensures that there is no correlation between the selected tuples for embedding (selected by $k1$) and the corresponding bit value positions in wm_data (selected by $k2$). They also suggest to perform embedding based on multiple categorical attributes by considering not only the association between the primary key and single categorical attribute A but all association between primary key and categorical attributes to increase robustness of the scheme. Although this scheme is claimed to be robust against serious attacks (*e.g.* random attacks), however, the

scheme is not suitable for database relations that need frequent updates, since it is very expensive to re-watermark the updated database relations. Though only a small part of selected tuples are affected by watermark embedding, the modifications of categorical attributes (*e.g.* change from "red" to "blue") in certain applications may be too significant to be acceptable. This watermarking technique is applied to binned medical data in a hierarchical manner.

2.1.3 Watermarking Based on Non-Numeric Multi-Word Attributes

This watermarking scheme which is based on hiding binary image in spaces of non-numeric multi-word attributes of subsets of tuples, instead of numeric attribute at bit-level. The watermark is divided into m string each containing n bits. On the other hand, the database is also divided into non-intersecting subsets each containing m tuples. The m short strings of the watermark image are embedded into each m-tuple subset. The embedding is done as follows: suppose the integer representation of the i_{th} , $i \in [1 \ldots m]$, short string is d_i. A double space is created after d_i words of the pre-selected nonnumeric, multi-word attribute of i_{th} tuple in the subset. The extraction phase counts the number single spaces appearing before double space which indicates the decimal equivalent of the embedded short binary string. Since the proposed algorithm embeds the same watermark for all non-intersecting subsets of the database, it is robust against subset deletion, subset addition, subset alteration and subset selection attacks. Another advantage for space- based watermarking is that large bit-capacity available for hiding the watermark which may also facilitate embedding of multiple small watermarks. However, it may suffer from watermark removal attack if Mallory replaces all double spaces between two words (if exist) by single space for all tuples in the relation.

2.1.4 Watermarking Based on Tuple or Attribute Insertion

(a) Fake tuples as watermark information

This approach aims to generate fake tuples and insert them erroneously into the database. The fake tuple creation algorithm take care of candidate key attributes and sensitivity level of non candidate attributes. He uses Bernoulli sampling probability p_i for the i_{th} non-candidate attribute A_i to decide its fake value which may be chosen uniformly or as the value with higher occurrence frequency in the existing set of values of A_i in the relation. Unlike other algorithms, the detection algorithm is not an inverse algorithm to the watermark generating algorithm and insertion algorithm is probabilistic in nature. Detection algorithm checks to see whether the fake tuples inserted during watermark insertion phase, exist or has been changed. It checks it via primary key. As soon as it finds one match (i.e. identical or similar tuples), detection is done. The detection will fail for the watermarked database when all of the fake tuples are deleted by benign deletions. The number of fake tuples to be inserted is decided by the database owner. However, the watermark insertion phase must take into account the fact that the values of the fake tuples marks should not by any means degrade the quality of the data in the database and should not impact the query results. One advantage of this scheme is that the ownership can be publicly verified more than once until all the fake tuples are revealed and the scheme does not suffer from incremental updatability.

(b) Virtual attribute as watermark information

Rather than inserting fake tuples, there is another watermarking technique can be used by inserting a virtual attribute in the relation which will serve as watermark containing parity checksum of all other attributes and an aggregate value obtained from any one of the numeric attribute of all tuples. The process of virtual attribute insertion is performed independently for each non-overlapping partitions obtained from the original relation. This scheme is designed

to authenticate the tamper-proof receipt of the database over an insecure communication channel. Although this approach is fragile and can easily detect any of the deletion or insertion or alter attacks, it suffers from the watermark removal attack.

2.2 Distortion-free Watermarking

Most of the distortion free watermarking techniques are fragile in the sense that in addition to the ownership claiming, they aim at maintaining the integrity of the information in the database. The watermark insertion phase does not depend on any specific type of attribute and does not introduce any distortion in the underlying data of the database.

2.2.1 Extracting Hash Value as Watermark Information

In order to achieve the purpose of fragile watermark, these watermarking schemes are able to detect any modifications made to a database relation. These schemes are designed for categorical data that cannot tolerate distortion, hence, the watermark embedding is distortion free. In the first one, partitioning of tuples is based on the hash value parameterized with primary key and secret key, whereas in the second one, partitioning is based on categorical attribute values. After partitioning, the tuple level and group level hash values for each group are computed. In the first one, a watermark of length equal to the number of tuple pairs in the group, is extracted from the group level hash value and for each tuple pair, the order of the two tuples are changed or unchanged according to their tuple hash values and the corresponding watermark bit. Moreover, it suggests to perform the exchange of tuples' positions based on Myrvold and Ruskeys linear permutation unranking algorithm to increase the embedding capacity. In these schemes, any modification of an attribute value will affect the watermarks in two groups as the modified tuple may be removed from one group and be added to the other group.

2.2.2 Combining Owner's Mark and Database Features as Watermark Information

This scheme aims at maintaining the integrity of the information in the database and is based on public authentication mechanism. The idea behind of this scheme is that, first an watermark W is created which is a $\sqrt{n} \times \sqrt{n}$ white image, where n is the no. of tuples in the relation, besides four corners having mark of the owner. It creates a value C_i ($0 \leq C_i \leq 255$) for each tuple t_i in the database using hash function MD5 and XOR operation. If there are n tuples in the database, it produces a feature

C of length n by combining all C_i in order. Finally, a certification code R is produced by XOR-ing C and W. The encrypted form of R using private key is made available publicly. During verification the integrity of the relation T, in similar way, it generates feature C from T. After decryption using public key the certification code R is XOR-ed with C that yield the watermark W. The integrity of this extracted watermark proves the integrity of the database.

2.2.3 Database Relation into Binary Form as Watermark Information

The public watermarking scheme by Li and Deng is applicable for marking any type of data including integer numeric, real numeric, character, and Boolean, without fear of any error constraints. The interesting features of this scheme is that it does not use any secret key and can be verified publicly as many times as necessary. The unique watermark key, used in both creation and verification phase, is public and obtained by one-way hashing from various information like the identity of the owner(s) and characteristics of the database (*e.g.* DB Name, Version etc.). Observe that the public watermark key is different from the public-private key pair of asymmetric cryptography. This watermark key is used to generate a watermark W from the relation R. The watermark W is a database relation whose schema is $W(P, W_0, ..., W_{\gamma-1})$, where $W_0, ..., W_{\gamma-1} \in \{0, 1\}$. Compared to database relation R, the watermark W has the same number η of tuples and the same primary key attribute P. The

number γ of binary attributes in W is a control parameter that determines the number ω of bits in W, where $\omega = \eta \times \gamma$ and $\gamma <$ *number of attributes* in R. In the algorithm, a cryptographic pseudorandom sequence generator (*e.g.*, Linear Feedback Shift Register) to randomize the order of the attributes and the MSBs of the attribute values are used for generating the watermark W. The use of MSBs is for thwarting potential attacks that modify the data. Since the watermark key K, the watermark W, and the algorithm are publicly known, anyone can locate those MSBs. Any modification to these MSBs introduces intolerable errors to the underlying data and can easily be captured during verification phase. However, alteration of other bits in the data cannot be detected by this scheme.

Another watermarking technique follows the same algorithmic steps as of the previous one. The basic difference is that the former considers a private key instead of public, and thus, cannot be publicly verifiable. In addition, the former is partition based and considers the extracted binary watermark as an image which is used to prove the ownership. This image is treated as the abstract counterpart of the concrete relation R, and the abstraction is sound in the sense that concretization of the abstract image must cover R. However, the disadvantage of this scheme is that the extracted image may not have any meaningful pattern.

It address the issue of persistency of watermarks, that serves as a way to recognize the integrity and ownership proof of the database while allowing the evaluation of the database by queries in a set of queries Q. The persistency of the watermark is preserved by exploiting the invariants (*Static Part* and *Semantics-based Properties* of the underlying data in the database *w.r.t.* Q) of the database states. The watermarking algorithms are designed as an improvement of the previous one in terms of fragileness and persistency.

2.2.4 R-tree Based Permutation as Watermark

In contrast to the traditional watermarking schemes, an R-tree data structure-based watermarking technique has been proposed. The proposed technique takes advantage of the fact that R-trees do not put conditions on the order of entries inside the node. In the proposed scheme, entries inside R-tree nodes are rearranged, relative to a secret initial order (a secret key), in a way that corresponds to the value of the watermark. To achieve that, they propose a one-to-one mapping between all possible permutations of entries in the R-tree node and all possible values of the watermark. Without loss of generality, watermarks are assumed to be numeric values. The proposed mapping employs a numbering system that uses variable base with factorial value. The detection rate of the malicious attacks depends on the nature of the attack, distribution of the data, and the size of the R-tree node. The proposed watermarking technique has the following desirable features:

(i) It does not change the values of the data in the R-tree node but rather hides the watermark in the relative order of entries inside the R-tree node.

(ii) It does not increase the size of the R-tree.

(iii) The proposed technique does not interfere with R-tree operations.

(iv) The performance overhead is minimal.

(v) The integrity check does not require the knowledge of unwatermarked data (blind watermark).

Chapter 3:
Proposed Approach

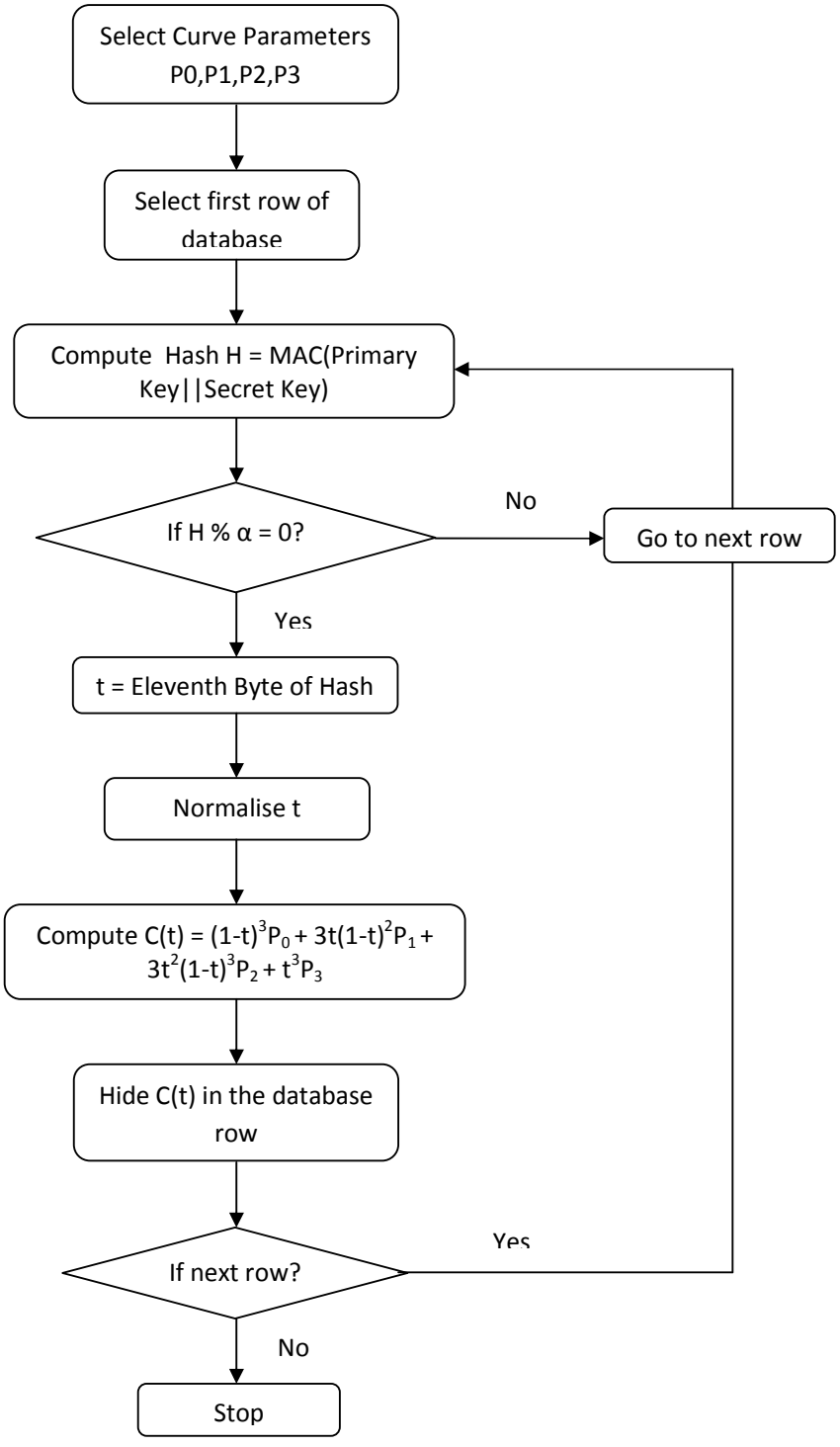
The proposed system creates a cubic curve according to the parameters provided and then hides the cubic curve in the database by using an embedding algorithm. Embedding algorithm utilizes a parameter alpha and a key, which provides control for the hiding and recovery processes, restricting extraction by those who do not possess the key, or do not have access to it.

3.1 Embedding Process

- The curve which is to be hidden in the database is selected according to four parameters: P_0, P_1, P_2 and P_3.
- The embedding process includes hiding the co-ordinates of the points lying on the cubic curve in the selected tuples in the database.
- The tuple is selected based on the secret key and value of alpha. For each tuple, primary key of that tuple and secret key are concatenated and its hash(H) is calculated. If the H mod α is zero then that tuple is selected to hide the curve points.
- Eleventh byte of the calculated hash value is normalised into a value between 0 and 1. This normalised value is denoted by 't' and is used to calculate the co-ordinates of a point lying on the curve. The value of 't' is put into the equation of the cubic curve which gives the x and y co-ordinates of a point lying on the curve:

$$C(t)=(1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)^3P_2 + t^3P_3$$

- First two attributes of the selected tuple- A1 and A2 are used to hide the x and y co-ordinates calculated above by replacing the decimal part of A1 and A2 with the x and y co-ordinates respectively.



(Figure 3.1: Embedding Process)

3.2 Extraction Process

- Four curve parameters: P_0, P_1, P_2 and P_3 which were used to select the curve during the embedding process are required to extract the hidden curve from the database.
- The tuple is selected based on the secret key and value of alpha. For each tuple, primary key of that tuple and secret key are concatenated and its hash(H) is calculated. If the H mod α is zero then that tuple has the hidden curve points.
- Then the first two attributes of the selected tuple- A1 and A2 are used to extract the hidden x and y co-ordinates from the decimal part of A1 and A2 respectively.
- Eleventh byte of the calculated hash value is normalised into a value between 0 and 1. This normalised value is denoted by 't' and is used to calculate the co-ordinates of a point lying on the curve. The value of 't' is put into the equation of the cubic curve which gives the x and y co-ordinates of a point lying on the curve:

$$C(t)=(1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)^3P_2 + t^3P_3$$

- Then the calculated curve points are compared with the extracted curve points. If they are same then this curve point is a match.

Chapter 4:
Results

4.1 Environmental Setup

The following configuration has been used while conducting the experiments:

Hardware Configuration

Processor	: Intel core i3
Processor Speed	: 1.90 GHz
Main Storage	: 6.00 GB
Hard Disk Capacity	: 500 GB

Software Configuration

Operating System	: Windows 8
Language used	: Java

4.2 Results for Database Watermarking

The scheme was applied on a sample database containing 10,000 records.

- Values of curve parameters used:

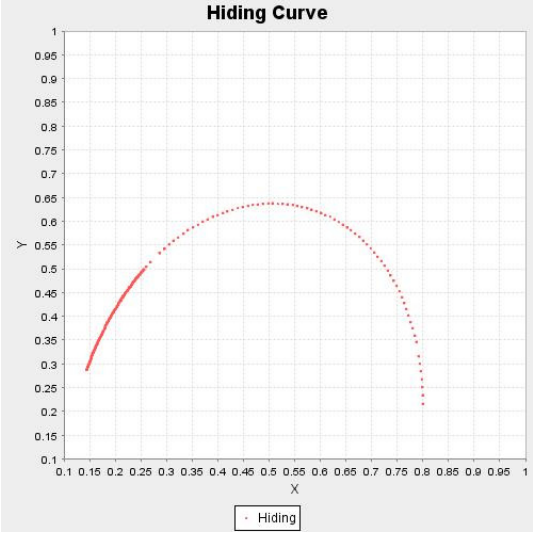
  $P_0 = (0.1,0.1)$

  $P_1 = (0.2,0.8)$

  $P_2 = (0.8,0.8)$

P_3 = (0.8,0.2)

- Secret Key = 1234
- Alpha = 10



(Figure 4.1: Graph showing watermark hidden in the database)

After hiding the watermark in the database, the database was subjected to modification, insertion and deletion of records.

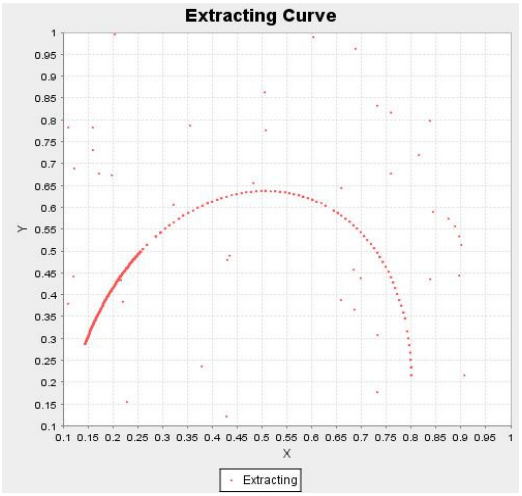
4.2.1 Results after records modification in database

Percentage of Records Modified	Total Curve Points Hidden	Number of Curve Points Extracted	Number of Curve Points Matched	Percentage of Curve Points Matched
5%	1023	1023	967	94.53%
10%	1023	1023	936	91.50%
15%	1023	1023	874	85.43%
20%	1023	1023	815	79.67%
25%	1023	1023	754	73.70%
30%	1023	1023	726	70.97%
35%	1023	1023	676	66.08%
40%	1023	1023	598	58.46%
45%	1023	1023	568	55.52%
50%	1023	1023	524	51.22%
55%	1023	1023	454	44.38%

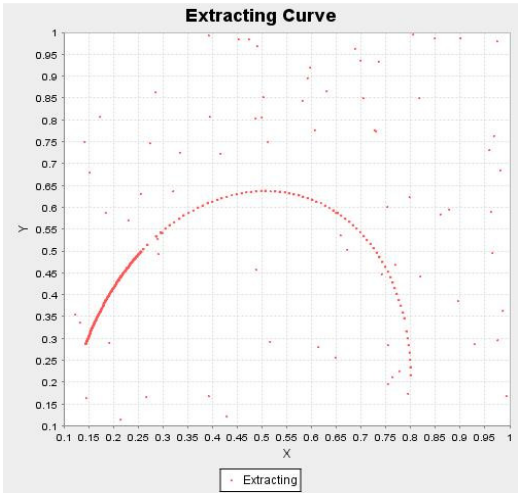
60%	1023	1023	381	37.24%
65%	1023	1023	380	37.15%
70%	1023	1023	309	30.21%
75%	1023	1023	264	25.81%
80%	1023	1023	199	19.45%
85%	1023	1023	168	16.42%
90%	1023	1023	101	9.87%
95%	1023	1023	45	4.40%

(Table 4.1: Percentage of Curve Points Matched after different degrees of records modification in database)

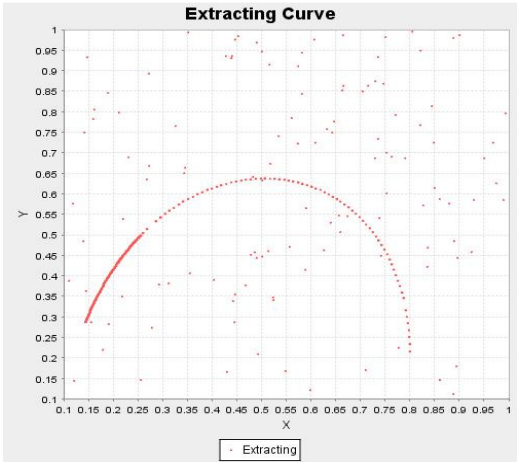
Graphs showing extracted watermark curves after different degrees of records modification in database



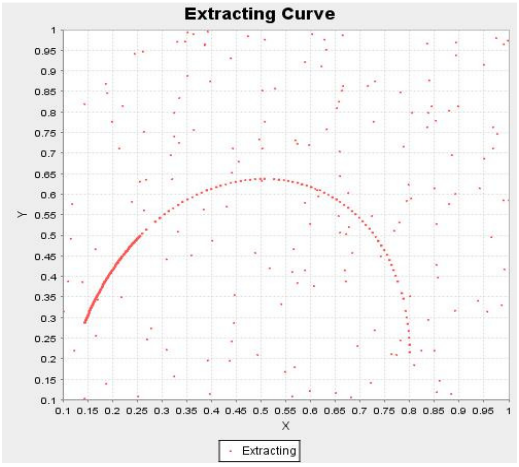
(a) 5% records modification



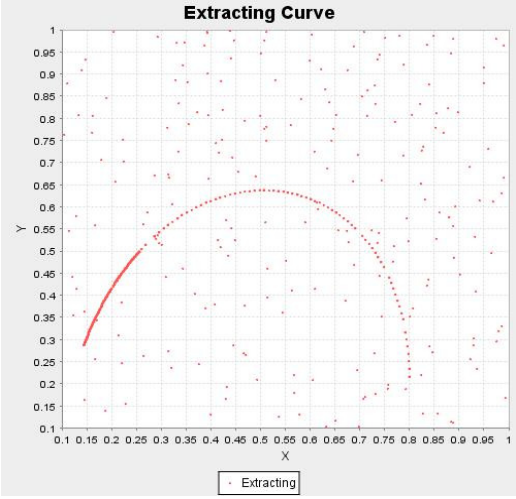
(b) 10% records modification



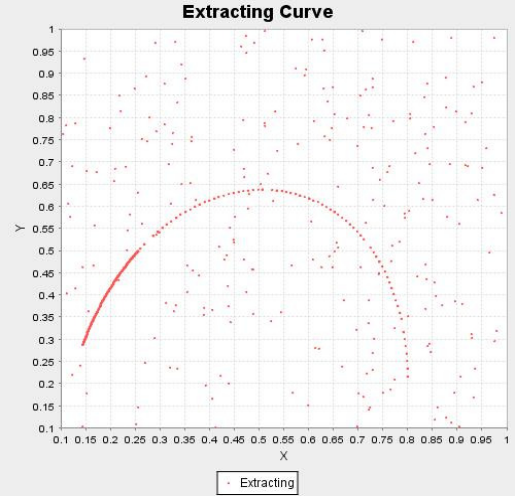
(c) 15% records modification



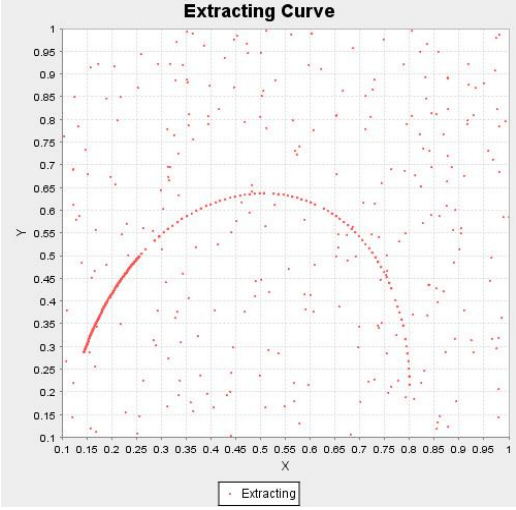
(d) 20% records modification



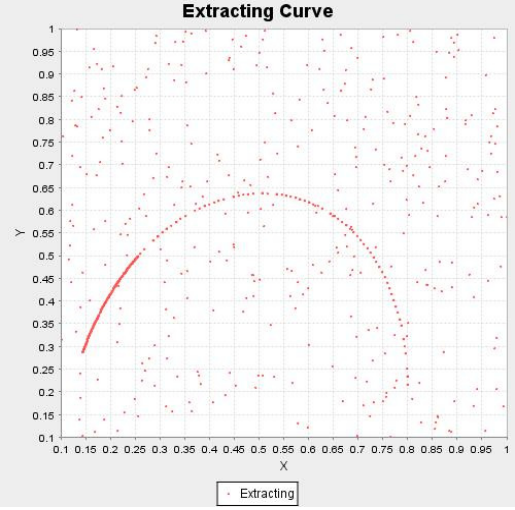
(e) 25% records modification



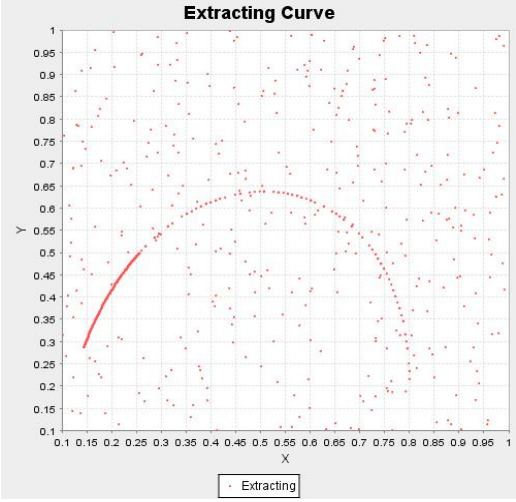
(f) 30% records modification



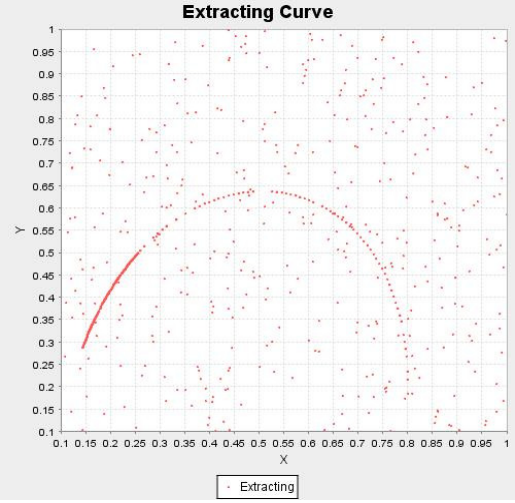
(g) 35% records modification



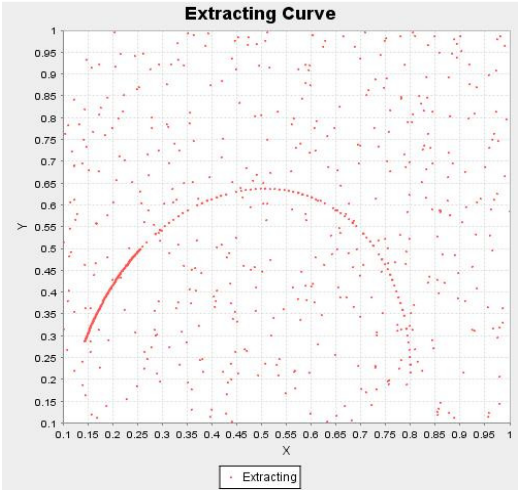
(h) 40% records modification



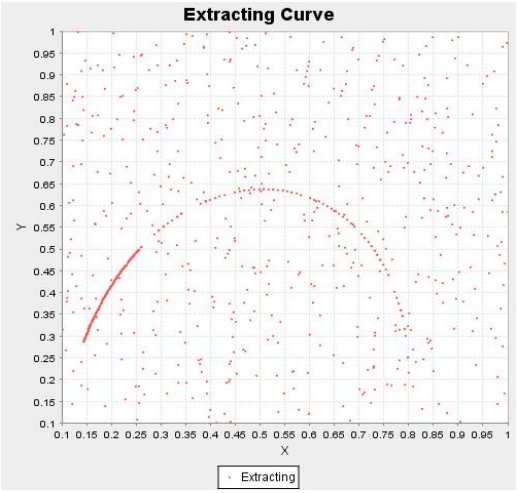
(i) 45% records modification



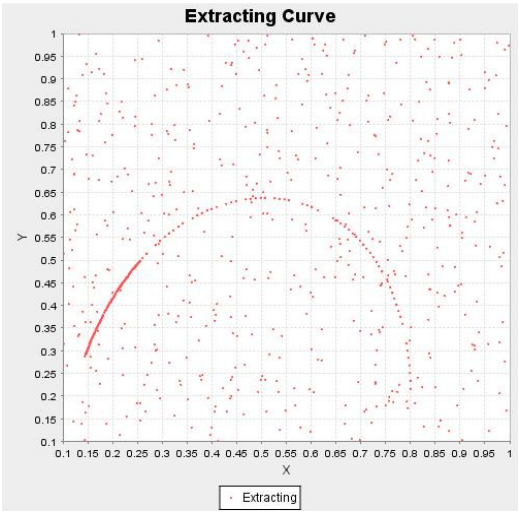
(j) 50% records modification



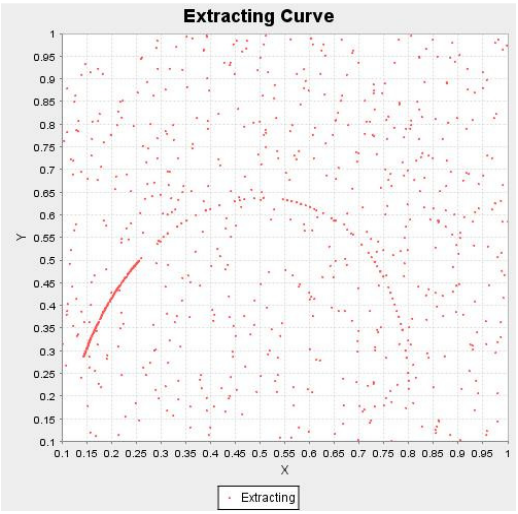
(k) 55% records modification



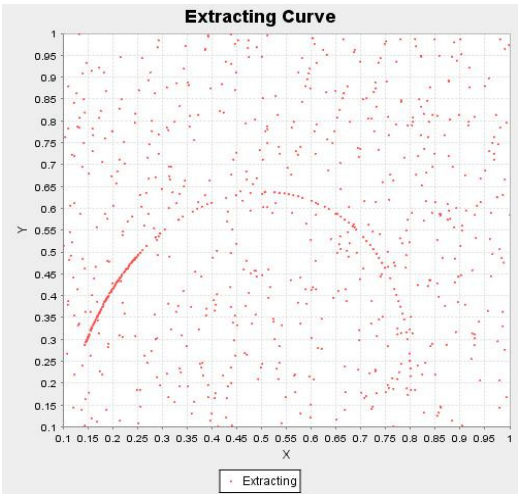
(l) 60% records modification



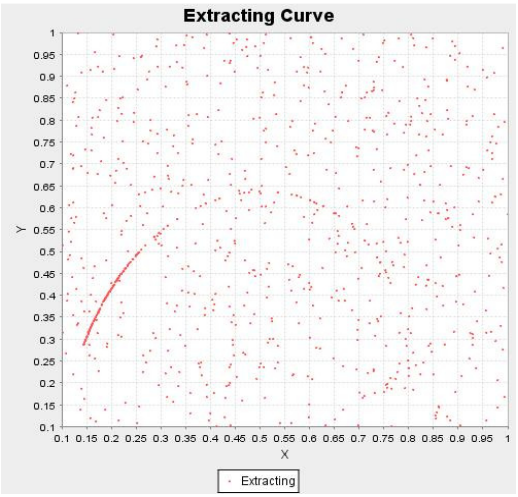
(m) 65% records modification



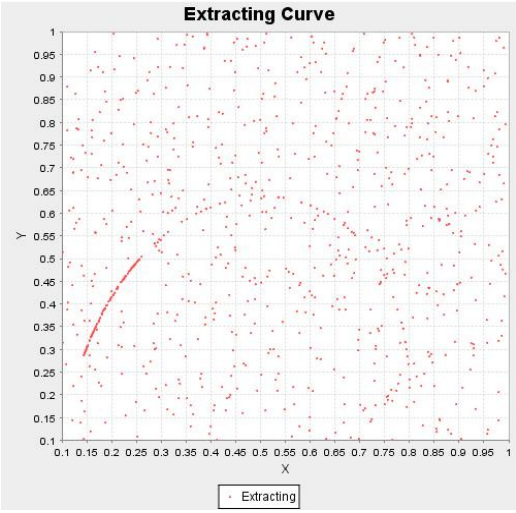
(n) 70% records modification



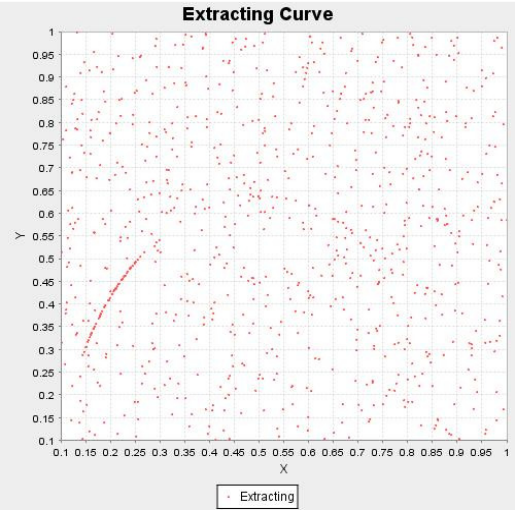
(o) 75% records modification



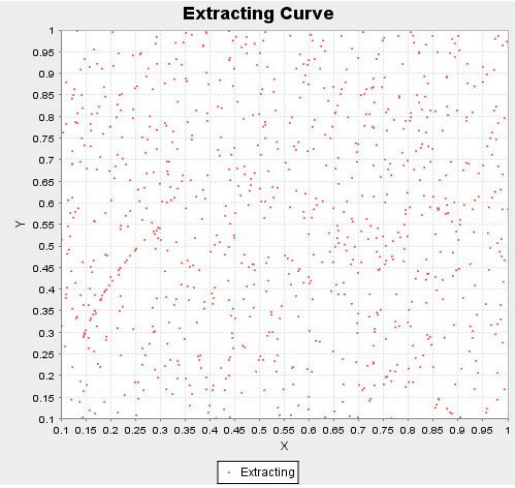
(p) 80% records modification



(q) 85% records modification



(r) 90% records modification



(s) 95% records modification

(Figure 4.2: Graphs showing extracted watermark curves after different degrees of records modification in database)

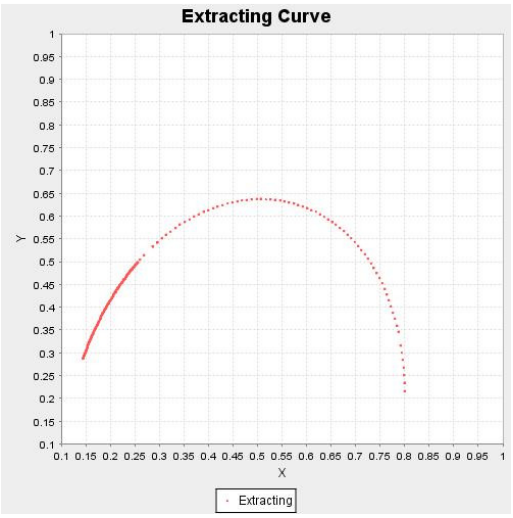
4.2.2 Results after records deletion in database

Percentage of Records Deleted	Total Curve Points Hidden	Number of Curve Points Extracted	Number of Curve Points Matched	Percentage of Curve Points Matched
5%	1023	1023	963	94.13%
10%	1023	1023	915	89.44%
15%	1023	1023	872	85.24%
20%	1023	1023	823	80.45%
25%	1023	1023	775	75.76%
30%	1023	1023	714	69.79%

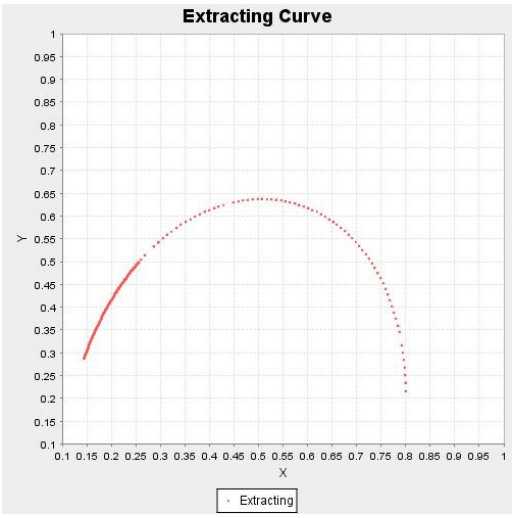
35%	1023	1023	646	63.15%
40%	1023	1023	622	60.80%
45%	1023	1023	530	51.81%
50%	1023	1023	511	49.95%
55%	1023	1023	453	44.28%
60%	1023	1023	402	39.30%
65%	1023	1023	344	33.63%
70%	1023	1023	286	27.96%
75%	1023	1023	262	25.61%
80%	1023	1023	210	20.53%
85%	1023	1023	154	15.05%
90%	1023	1023	109	10.65%
95%	1023	1023	52	5.08%

(Table 4.2: Percentage of Curve Points Matched after different degrees of records deletion in database)

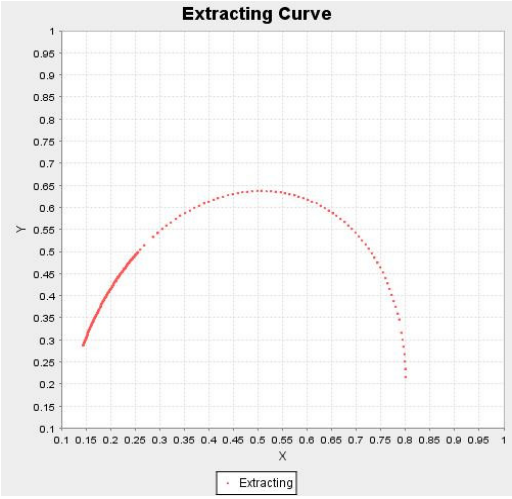
Graphs showing extracted watermark curves after different degrees of records deletion in database



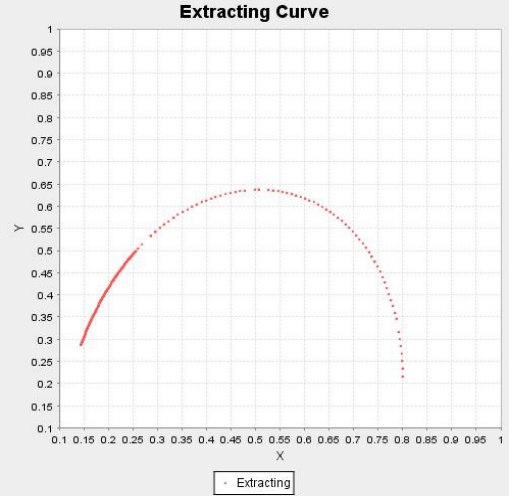
(a) 5% records deletion



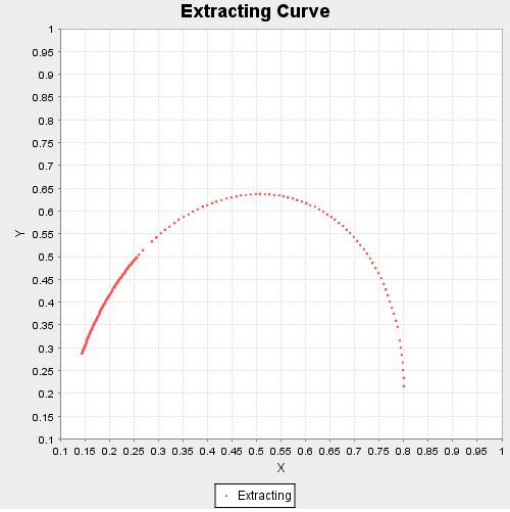
(b) 10% records deletion



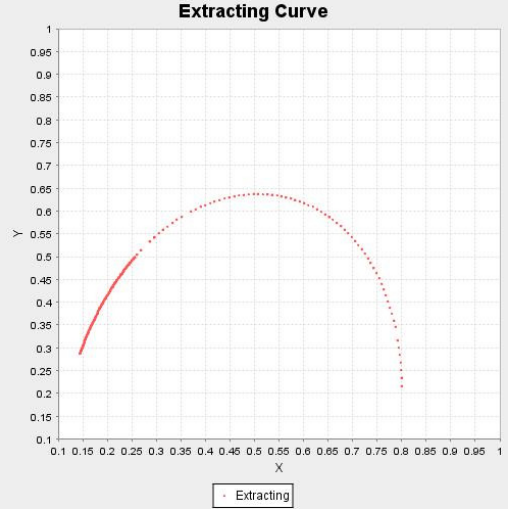
(c) 15% records deletion



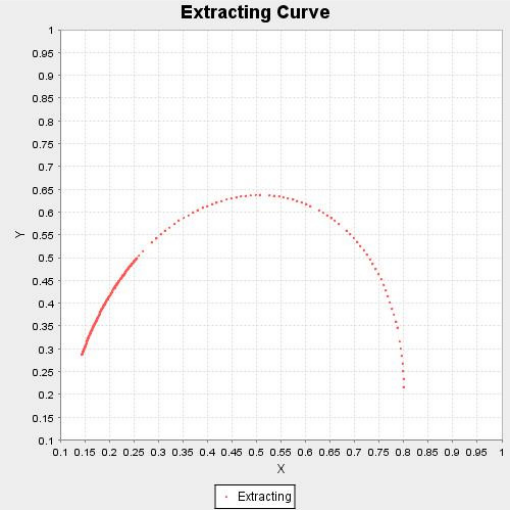
(d) 20% records deletion



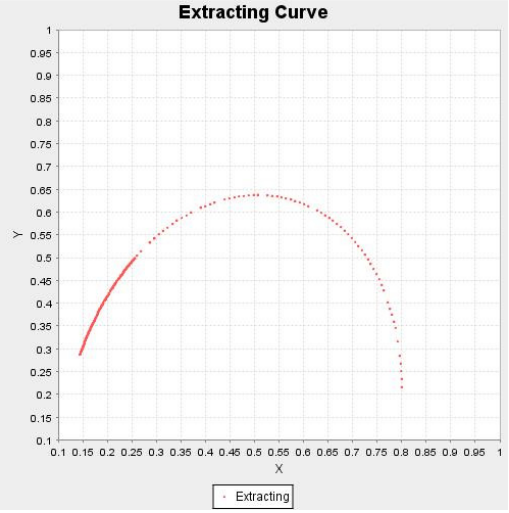
(e) 25% records deletion



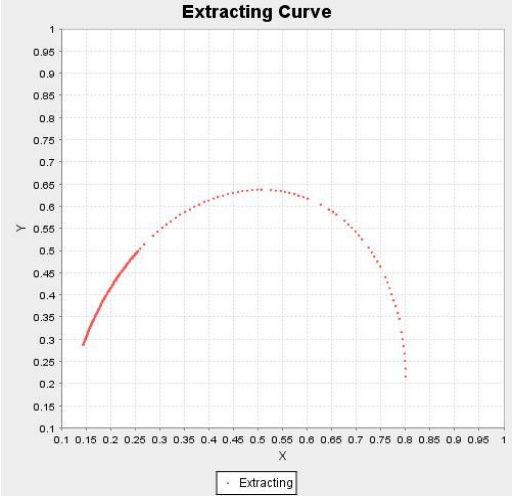
(f) 30% records deletion



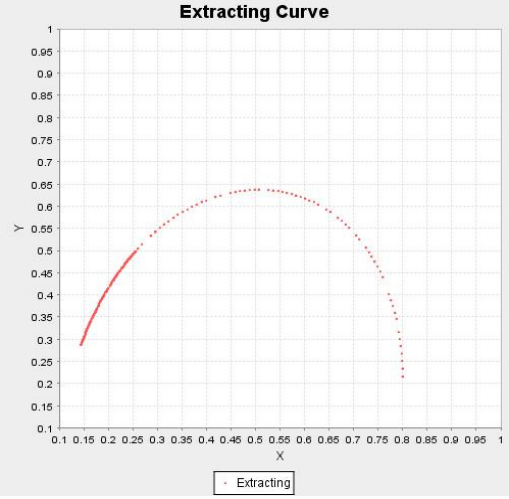
(g) 35% records deletion



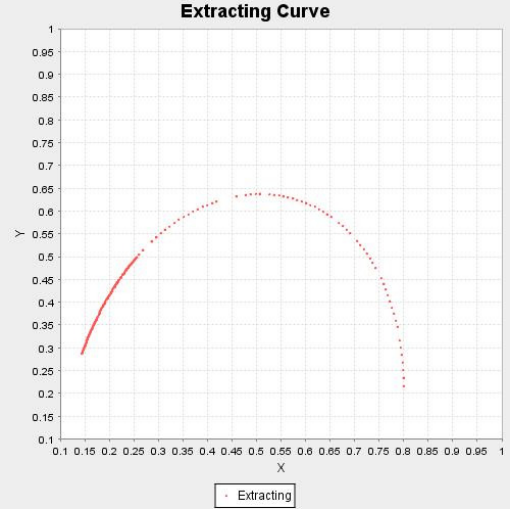
(h) 40% records deletion



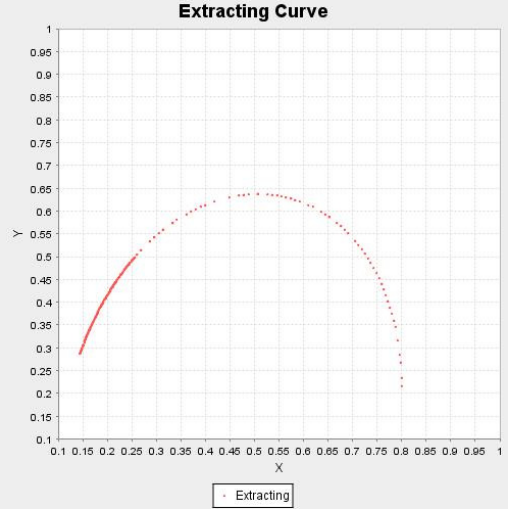
(i) 45% records deletion



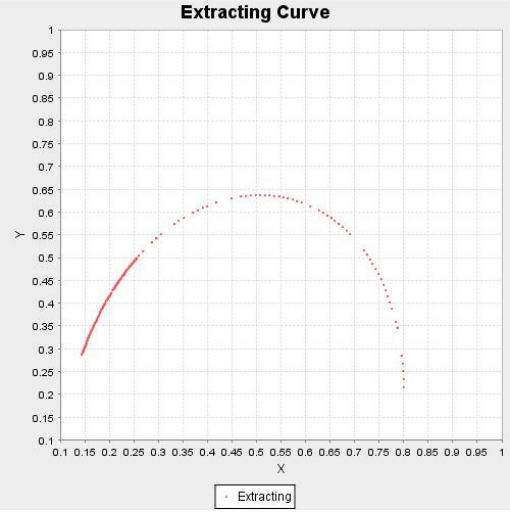
(j) 50% records deletion



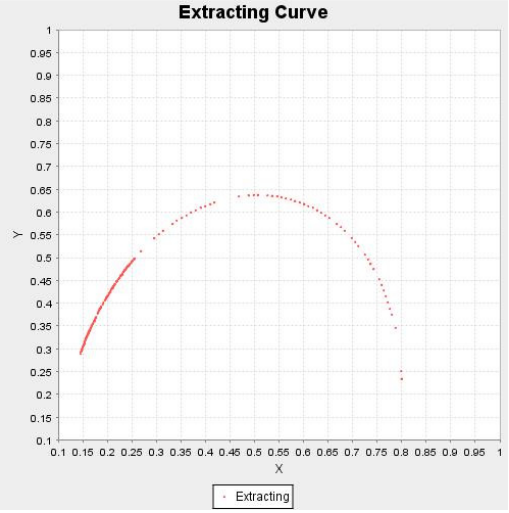
(k) 55% records deletion



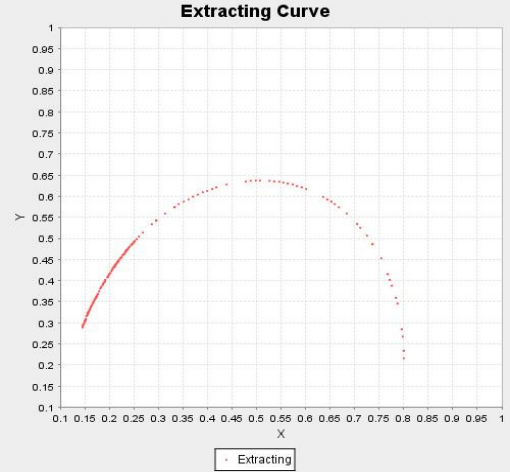
(l) 60% records deletion



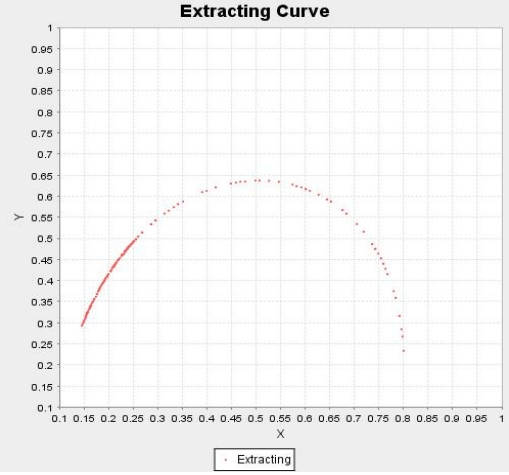
(m) 65% records deletion



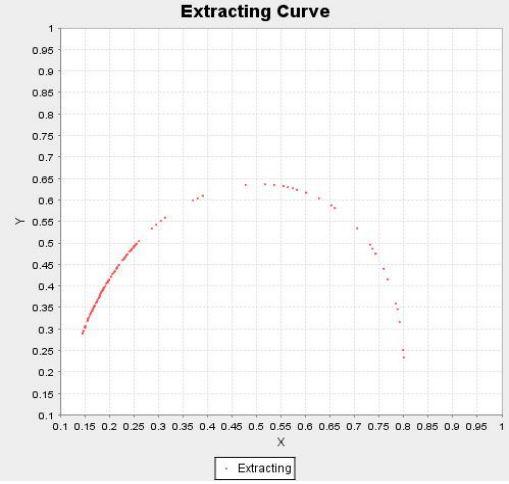
(n) 70% records deletion



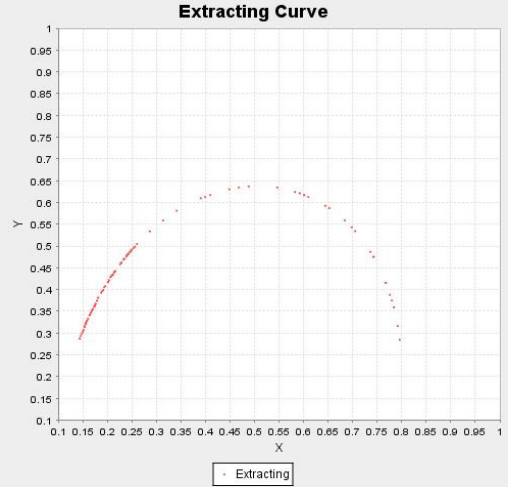
(o) 75% records deletion



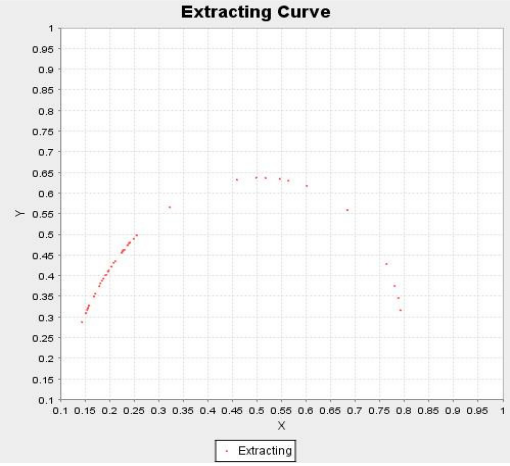
(p) 80% records deletion



(q) 85% records deletion



(r) 90% records deletion



(s) 95% records deletion

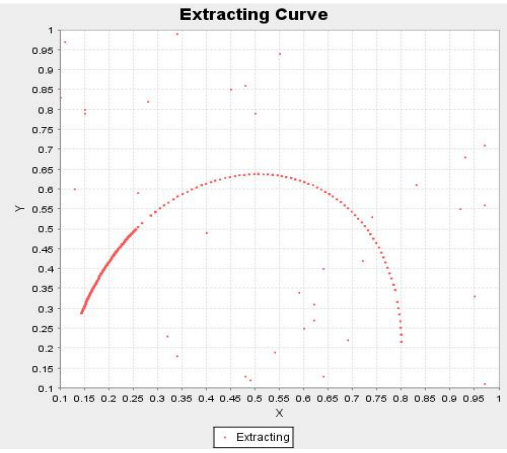
(Figure 4.3: Graphs showing extracted watermark curves after different degrees of records deletion in database)

4.2.3 Results after records insertion in database

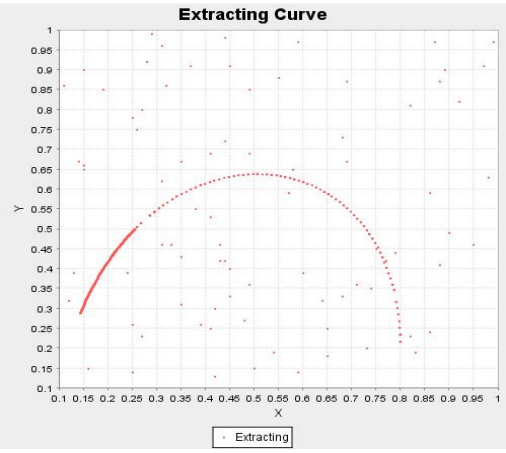
Percentage of Records Inserted	Total Curve Points Hidden	Number of Curve Points Extracted	Number of Curve Points Matched	Percentage of Curve Points Matched
5%	1023	1074	1023	95.25%
10%	1023	1126	1023	90.85%
15%	1023	1177	1023	86.92%
20%	1023	1212	1023	84.41%
25%	1023	1255	1023	81.51%
30%	1023	1313	1023	77.91%
35%	1023	1359	1023	75.28%
40%	1023	1408	1023	72.66%
45%	1023	1456	1023	70.26%
50%	1023	1503	1023	68.06%
55%	1023	1549	1023	66.04%
60%	1023	1606	1023	63.70%
65%	1023	1654	1023	61.85%
70%	1023	1707	1023	59.93%
75%	1023	1768	1023	57.86%
80%	1023	1818	1023	56.27%
85%	1023	1873	1023	54.62%
90%	1023	1916	1023	53.39%
95%	1023	1971	1023	51.90%

(Table 4.3: Percentage of Curve Points Matched after different degrees of records insertion in database)

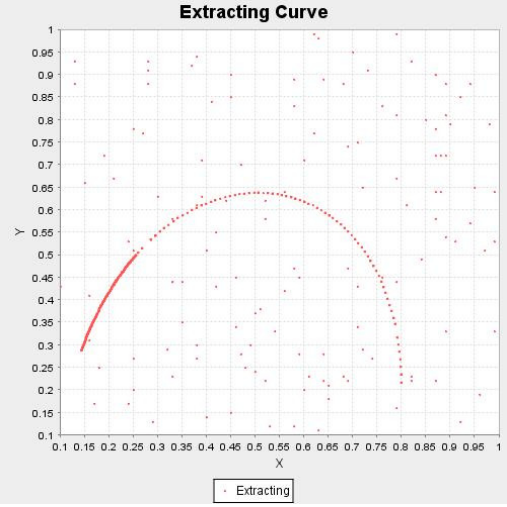
Graphs showing extracted watermark curves after different degrees of records insertion in database



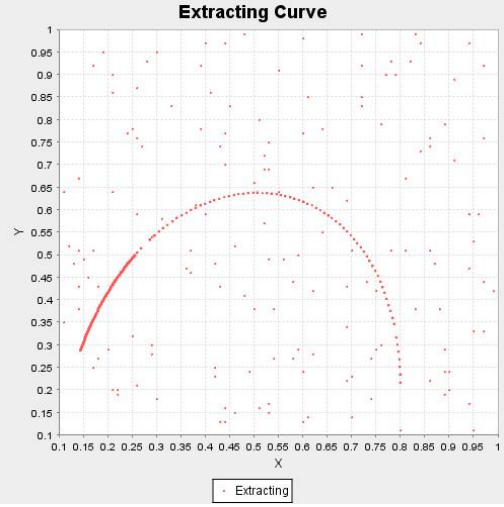
(a) 5% records insertion



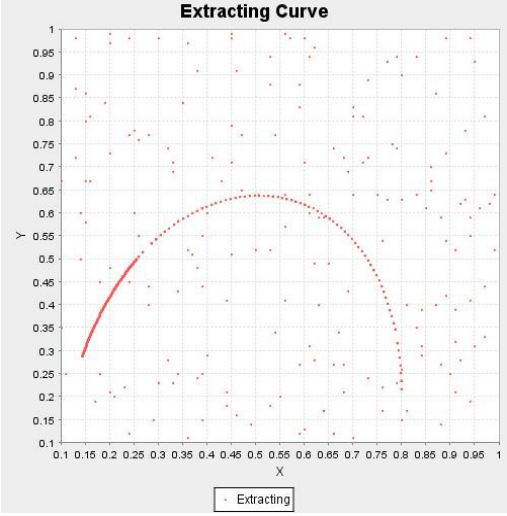
(b) 10% records insertion



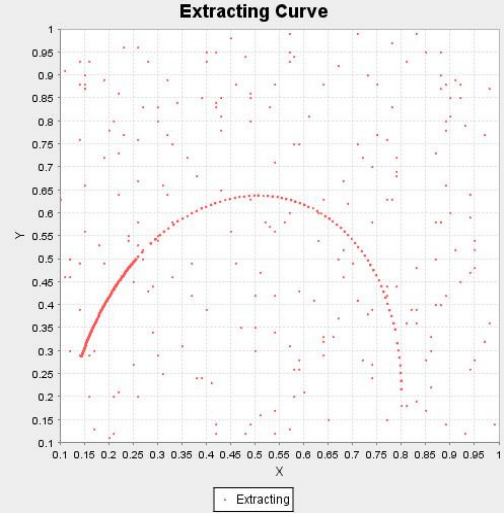
(c) 15% records insertion



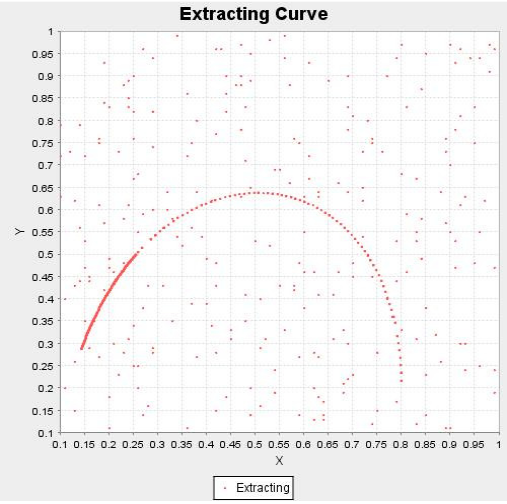
(d) 20% records insertion



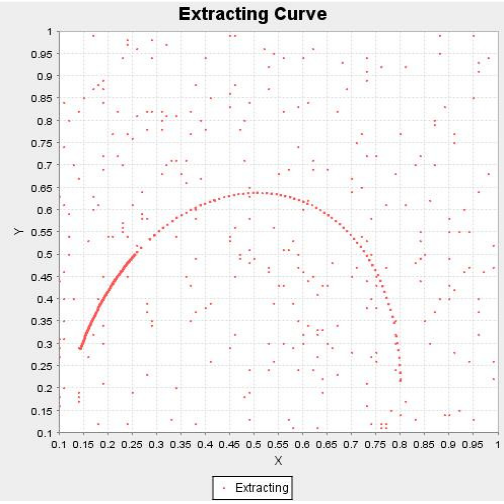
(e) 25% records insertion



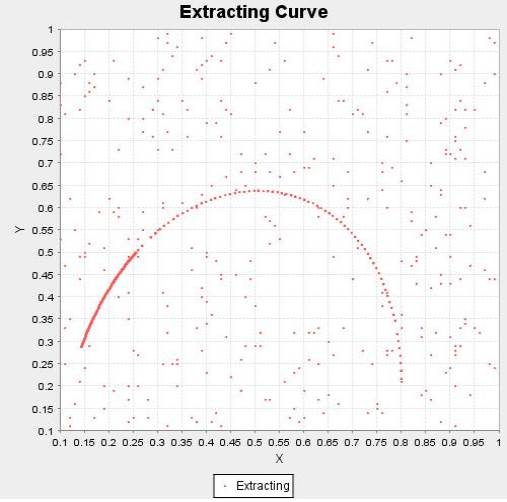
(f) 30% records insertion



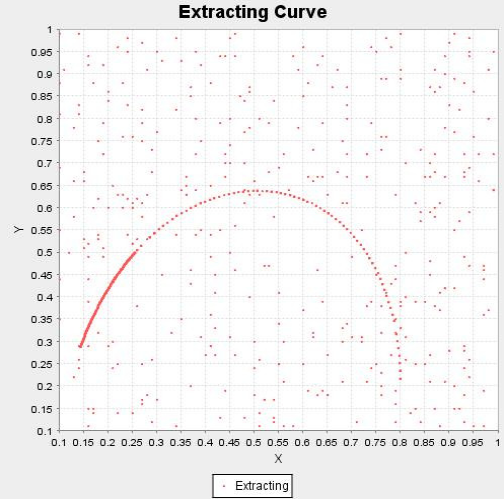
(g) 35% records insertion



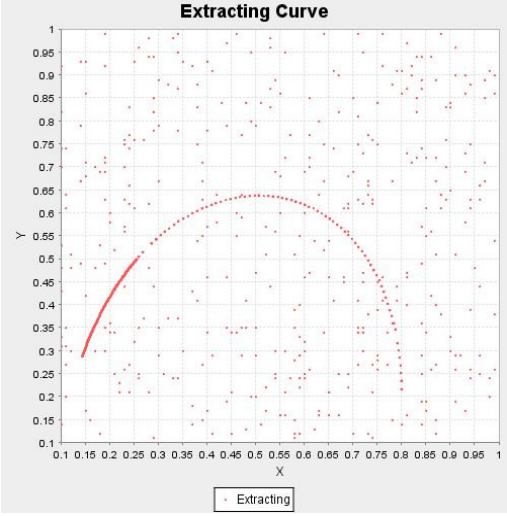
(f) 40% records insertion



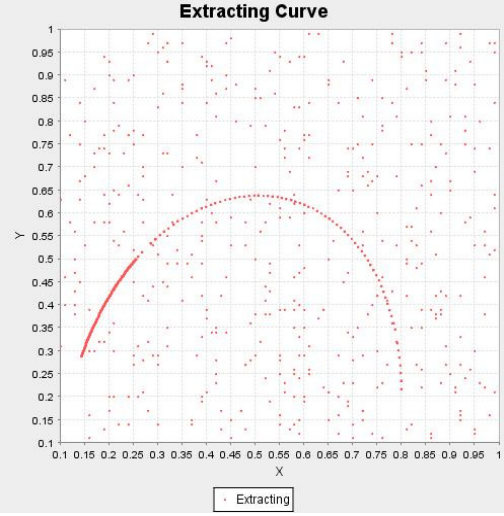
(i) 45% records insertion



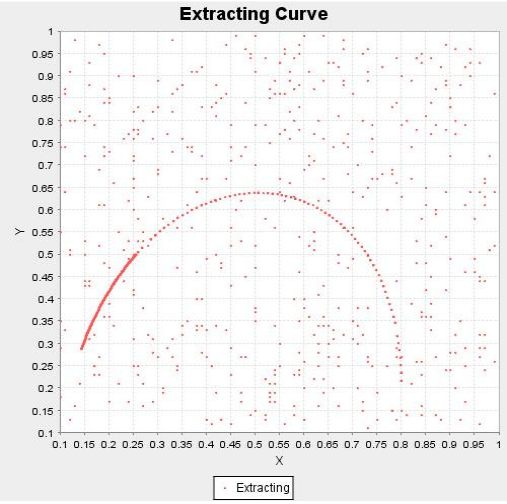
(j) 50% records insertion



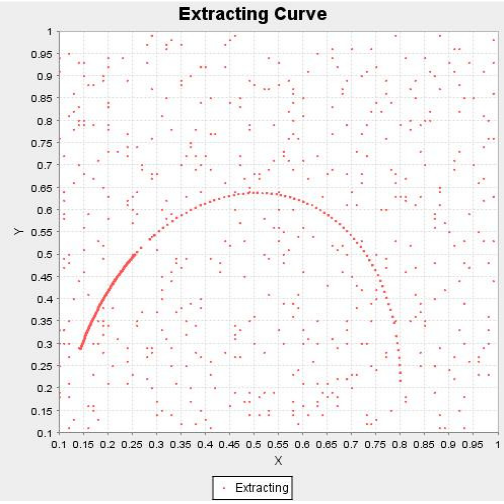
(k) 55% records insertion



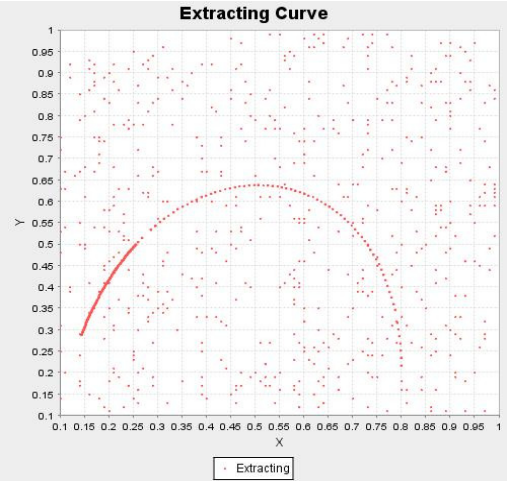
(l) 60% records insertion



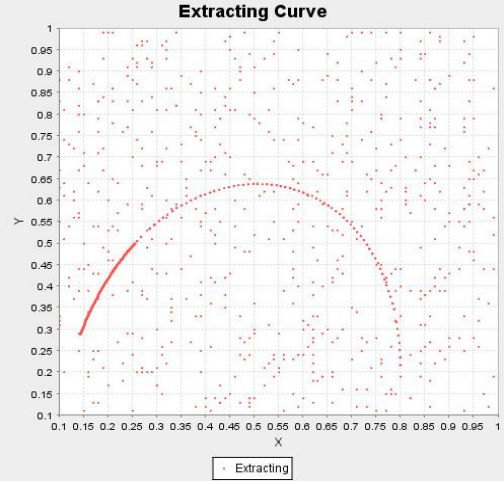
(m) 65% records insertion



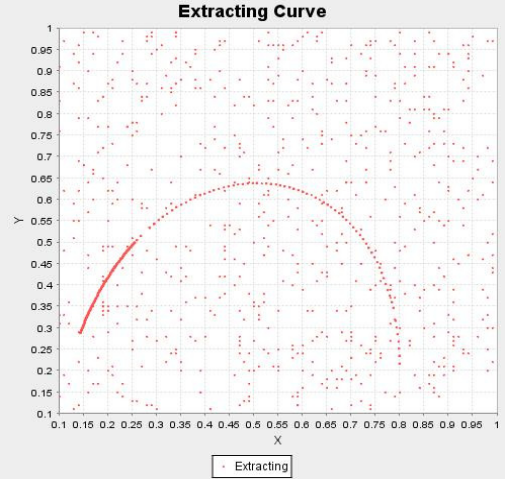
(n) 70% records insertion



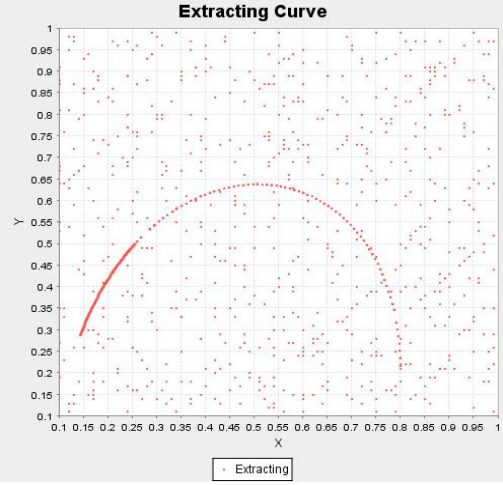
(o) 75% records insertion



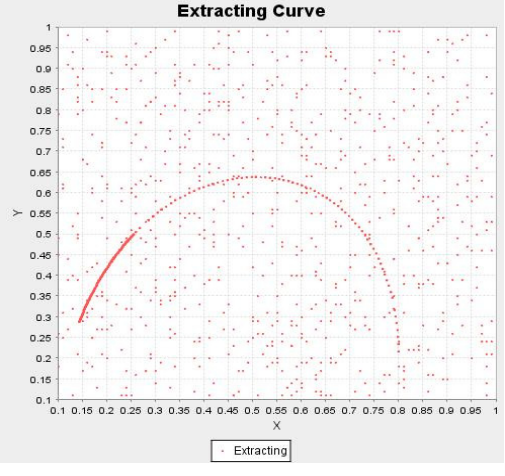
(p) 80% records insertion



(q) 85% records insertion



(r) 90% records insertion



(s) 95% records insertion

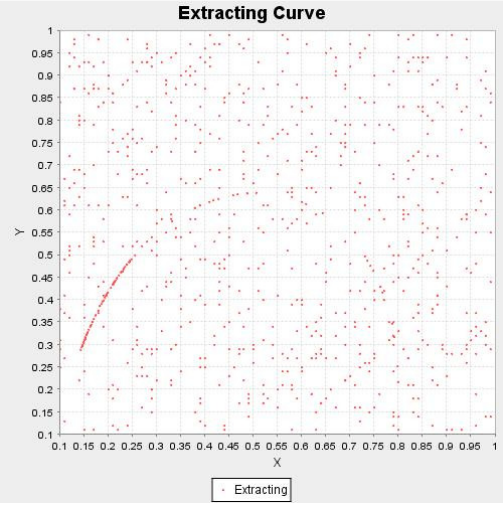
(Figure 4.4: Graphs showing extracted watermark curves after different degrees of records insertion in database)

4.2.4 Result if using the incorrect secret key

Secret Key used for Watermarking = 1234

Secret Key used for Extracting the Watermark = 1235

Change of one lowest significant bit causes significant change in the extracted watermark.



(Figure 4.5: Graph showing extracted watermark curve if using incorrect key)

Chapter 5: Conclusion

We presented a new approach using Bezier curves and discussed the insertion and extraction watermarking algorithms in details. Our goal has been to ensure that our watermarking approach is robust for watermarking relational data. While we discussed our approach's robustness analytically, we did not evaluate the approach quantitively. In fact, evaluating watermarks for relational database is a challenge and requires further consideration. However, the persistency of the watermark after both malicious and benign updates, as a subproblem, might be evaluated by acquiring access to a log of user queries on a particular database over a reasonably long period of time, and then run the log on the watermarked database and observe whether the watermark detection algorithm will confirm the watermark. While this evaluation process sounds plausible, it is application-specific and may not be generalized very well.

References

[1] R. Chamlawi, A. Khan, I. Usman, “Authentication and recovery of images usingmultiple watermarks”, Computers & Electrical Engineering 36 (2010), 578–584.

[2] R. Sion, M. Atallah and S.Prabhakar, “Rights Protection for Relational Data”, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no.12, (2004), pp. 1509-1525.

[3] I. Cox, M. Miller, J. Bloom and C. Honsinger, “Digital Watermarking”, Academic Press, USA (2002).

[4] Khurram Jawad, Asifullah Khan, "Genetic algorithm and difference expansion based reversiblewatermarking for relational databases", The Journal of Systems and Software 86 (2013) 2742– 2753.

[5] Zhiyong Li, Junmin Liu and Weicheng Tao, “A Novel Relational Database Watermarking Algorithm Based on Clustering and Polar Angle Expansion”, International Journal of Security and Its Applications, Vol. 7, No. 2, March, 2013.

[6] R. Agrawal and J. Kiernan, “Watermarking Relational Databases”, Proc. VLDB’02, (2002), pp. 155-166.

[7] R. Sion, M. Atallah and S. Prabhakar, “On Watermarking Numeric Sets”, Proc. IWDW, (2002), pp. 12-15.

[8] X. Niu, et al., “Watermarking Relational Databases for Ownership Protection”, Chinese Journal of Electronics (in Chinese), vol. 31, no. 12A, (2003), pp. 2050-2053.

[9] Y. J. Li, V. Swarup and S. Jajodia, "Fingerprinting Relational Databases: Schemes and Specialties", IEEE Transactions on Dependable Secure Computing, vol. 2, no. 1, (2005), pp. 34-45.

[10] Y. Zhang, X. M. Niu and D. N. Zhao, "A Method of Protecting Relational Databases Copyright with Cloud Watermark", Proc. World Academy of Science, Engineering and Technology, vol. 3, (2005), pp. 68-72.

[11] Y. Zhang, B. Yang and X. M. Niu, "Reversible Watermarking for Relational Database Authentication", Journal of Computers, vol. 17, no. 2, (2006), pp. 59-65.

[12] G. Gupta and J. Pieprzyk, "Reversible and Blind Database Watermarking Using Difference Expansion", International Journal of Digital Crime and Forensics, vol. 1, no. 2, (2009), pp. 42-54.

[13] Z. H. Zhang, et al., "Watermarking Relational Database Using Image", Proc. ICMLC, (2004), pp. 1739-1744.

[14] X. M. Jin, et al., "Watermarking spatial trajectory database", Proc. DASFAA, (2005), pp. 56-67.

[15] X. C. Cui, et al., "A Robust Algorithm for Watermark Numeric Relational Databases", Intelligent Control and Automation, vol. 344, (2006), pp. 810-815.

[16] H. P. Guo, et al., "A Fragile Watermarking Scheme for Detecting Malicious Modifications of Database Relations", Information Sciences, vol. 176, no. 10, (2006), pp. 1350-1378.

[17] M. L. Meng, X. C. Cui and H. Cui, "The Approach for Optimization in Watermark Signal of Relational Databases by using Genetic Algorithms", Proc. ICCSIT, (2008), pp. 448-452.

[18] I. Kamel, "A Schema for Protecting the Integrity of Databases", Computers & Security, vol. 28, no. 7, (2009), pp. 698-709.

[19] M. Shehab, E. Bertino and A. Ghafoor, "Watermarking Relational Databases Using Optimization-based Techniques", IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 1, (2008), pp. 116-129.

[20] G. Gupta and J. Pieprzyk, "Database Relation Watermarking Resilient against Secondary Watermarking Attacks", Proc. 5th International Conference on Information Systems Security, (2009), pp. 222-236.

[21] S. Bhattacharya and A. Cortesi, "A Generic Distortion Free Watermarking Technique for Relational Databases", Proc. 5th International Conference on Information Systems Security, (2009), pp. 252-264.

[22] A. H. Ali, et al., "Copyright Protection of Relational Database Systems", Proc. 2nd International Conference on Networked Digital Technologies, vol. 87, (2010), pp. 143-150.

[23] G. A. David, "Query-Preserving Watermarking of Relational Databases and XML Documents", ACM Transactions on Database System, vol. 36, no. 1, (2011), pp. 301-324.

[24] M. E. Farfour, et al., "A blind reversible method for watermarking relational databases based on a time-stamping protocol", vol. 39, no. 3, (2012), pp. 3185-3196.

[25] Raju Halder, Shantanu Pal, Agostino Cortesi, "Watermarking Techniques for Relational Databases:Survey, Classification and Comparison", Journal of Universal Computer Science, vol. 16, no. 21 (2010), 3164-3190.

[26] Agrawal, R., Haas, P. J., and Kiernan, J. (2003a), "A system for watermarking relational databases", ACM SIGMOD International Conference on Management of data (SIGMOD '03).

[27] Agrawal, R., Haas, P. J., and Kiernan, J. (2003b), "Watermarking relational data: framework, algorithms and analysis", The VLDB Journal, 12:157–169.