WATERMARKING OF RELATIONAL DATABASES

YAXIAO SONG

NATIONAL UNIVERSITY OF SINGAPORE

2004

WATERMARKING OF RELATIONAL DATABASES

YAXIAO SONG

(B.E, HARBIN INSTITUTE OF TECHNOLOGY,CHINA)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgement

I would like to thank Prof. Ooi Beng Chin and Prof. Mohan S. Kankanhalli, my supervisors, for their many suggestions and constant support during my research. I am thankful to them for their guidance through the early stages of chaos and confusion. They encouraged me and helped me move on when I met difficulties in the work. Professor Nasir Memon expressed his interest in my work and gave me some inspirations. Minghao Cui shared with me his knowledge of digital watermarking and provided many useful references and friendly encouragement. I should thank National University of Singapore who supported my first year graduate study, and the research project of Location Based System that support my remaining study. I'm grateful to Prof. Huang Zhiyong and Prof. Huang Bo, who are the Vice PIs of the project, because their support and understanding make my research like this possible. I thank my friends who provide me with some valuable source codes and test data, which were crucial to the successful completion of this project.

I am grateful to my parents for their patience and love over the years, and my boyfriend, Pan Feng, for his support and cares and understanding. Without them this work would never have come into what it is today.

Finally, I wish to thank the following: Shichang (for his nagging); Liqi (for her friendship); Sky, Lisheng and Vicky (for special); Dan Lin, Shuai cheng, Yang Rui, Yao Zhen, and all my friends (for the good and bad times we had together).

Yaxiao Song

December 12, 2003

Abstract

Due to increasing concerns about data piracy, digital rights management of databases is becoming an extremely important area of research. There have recently been some pioneering works in this area that help establish ownership of relational data. In this thesis, we firstly consider the more general buyer-seller scenario where an owner of the database sells it to many different customers. For this case, obviously it is necessary to individually identify each copy of the data sold so as to precisely and reliably identify the source in case of piracy. Also, such a scheme would be susceptible to collusion attacks in which a set of legitimate buyers collude together to illegally tamper the watermark. We present a novel *Individualized Watermarking* scheme for the need. Secondly, we propose an *Invertible Watermarking* scheme for protecting precision-critical databases, where there is a need to reverse the original data. Experiments conducted on a commercial database system confirm that the proposed methods can survive a wide variety of attacks. Moreover, *Incremental Watermarking* for the second scenario is very effective when normal updates of the databases are frequently conducted.

Keywords: Watermarking of relational databases; Fingerprinting; Collusion-resistant watermarking; Collusion attacks; Invertible (Lossless, Reversible) watermarking

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid growth of the Internet, unauthorized duplication and distribution of digital data is becoming effortless requiring minimal capital expenditure and providing easy illegal revenue. Digital data is easy to replicate, alter and transmit. However, if such piracy becomes widespread, it would undermine the very basis of the trade of digital goods. These very characteristics that have enabled the digital revolution become a scourge from the protection point of view of intellectual property rights. Nowadays, not only image and music (audio) industries, but also movie (video) industries are encountering big piracy problems, due to the file-sharing Web communities and the increase of the Internet bandwidth. Therefore, digital rights management is increasingly being a matter of great concern.

Digital watermarking and fingerprinting appear to be very promising solutions for these problems. Digital watermarking aims at protecting a digital content from unauthorized redistribution and copying by enabling ownership provability over the content. Fingerprinting is a special form of digital watermarking for the purpose of identifying the recipients who have been provided the content. In case of unauthorized disclosure, the owner should be able to trace the source of piracy.

Though various techniques of digital watermarking and fingerprinting do not strictly prevent illegal copying, they do deter such copying by establishing the original ownership of an illegally redistributed copy. There is a rich body of literature on watermarking multimedia data [17, 7, 20]. Most of the research in the area of digital watermarking has focused on multimedia content which includes digital still images, audio and video sources. These techniques have traditionally relied on the availability of a large "bandwidth" within which information can be indelibly and imperceptibly inserted while remaining certain essential properties of the original contents. Much of the bandwidth is due to the insensitivity of human sensory system (for example, human visual system) to perceive the small changes or distortions introduced into the content.

More recently, with the focus of protecting digital rights shifting, there emerged new concerns on protecting the ownership of software [27], natural languages [3], digital circuits [23], and structured data like trees, graphs, or solution of optimization problems [21], etc.

In general, the goal of digital watermarking is to insert a robust watermark into the digital content such that the mark does not destroy the value of the content, and the mark is hard to be removed by adversaries without destroying the utility of content. The measurement of the value of the content is closely related to the data type and its intended use. Obviously, for images and video clips, the utility is judged by the visual quality of the images and video clips. As for text, the value may be in conveying the same meaning, while for software the value may be preserved by ensuring equivalent computation [27], etc.

The extensive use of databases in applications is creating a need for protecting copyright of databases. Although work in the area of database security is well-established [4, 5, 16, 18, 26], relatively little work has been done in the area of database watermarking. There have recently been some pioneering works reported in [2, 29] and the more recent work of [30]. There are a range of watermarking techniques available for protection of ownership, authentication and content integrity. Given the lead that multimedia watermarking research has over database watermarking research, one would hope that the standard multimedia techniques can be just carried over to the realm of relational databases. However, there exist some fundamental differences between the characteristics of multimedia data and relational data, which make the adaptation of the known watermarking techniques not as easy as one would have desired. The main differences between multimedia data and relational data from the point of view of watermarking are as following:

- Portions of multimedia object cannot be dropped freely [2]. On the contrary, database tuples are frequently deleted and inserted.
- Multimedia objects have fixed relative spatial/temporal positioning and neighborhood correlation, while tuples of a relation have no such implied ordering.
- Multimedia objects possess a tremendous amount of redundancy, thus providing a larger channel to hide information [2].
- Database tuples have primary and foreign key relationships, so that improper modification of an attribute value may not only destroy the usability of this value, but could destroy the result of join operation. On the contrary, with multimedia data, we do not have such concern.
- There are many psycho-physical phenomena based on the human visual system and human auditory system which can be exploited for watermark embedding. For example, the textured areas in images can be utilized for hiding many

watermark bits that cannot be noticed by the human eyes [19]. In general, one cannot exploit such phenomena in relational databases.

Due to the above challenges, techniques developed for multimedia data cannot be directly used for watermarking relational databases. Therefore, new watermarking techniques for databases have to be designed.

1.1 Motivation

Given the work and techniques devised for watermarking relational databases, it is now possible to establish ownership of a redistributed copy of relations. However, it is far inadequate to trace the illegal parties (traitors) who redistribute the database. The feasibility and reliability of traitor tracing is important as it helps identify the data buyer should the copyright issue be brought to court. Thus it is highly desirable if we can identify the recipients (buyers) to whom the data have been provided by the database owners. While techniques for fingerprinting multimedia data are well-established in literature [6, 10], they are not directly applicable to relational databases. In this thesis, we develop an *Individualized Watermarking* technique for relational databases that can both establish ownership and identify individual buyers of the relational databases. Besides surviving the attacks in the form of data re-sorting, subset selection, subset addition, etc, the individualized watermarks should be buyer specific and resistant to collusion attacks.

Besides the importance of traitor tracing, we are also motivated by the scenario where the integrity and accuracy of the data (precision-critical data) is as important as the ability to assert ownership and to trace traitors. Examples of real world data sets that are precision-critical include medical data, military data, satellite data, and so on. These data are both sensitive (where protection of the data is important) and precision-critical (where errors in the data renders the data useless). It is noteworthy that even a normal attribute such as *salary* can be sensitive and precision-critical, as *salary* is confidential while inaccuracy of this attribute is disastrous. Therefore, we present a technique of *Invertible Watermarking* databases, where privileged users of the databases can losslessly remove the watermark and view the intact data, while the non-privileged users can only get access to the watermarked data.

1.2 Problem Statement

The scope of this work falls into two major parts: one is *Individualized Watermarking* of relational databases, where ownership establishment and traitor tracing are enabled. The proposed technique is collusion-resistant, such that even the attempt of a few buyers colluding together to destroy the individualized watermarks will not be successful. The formal security analysis is given as well as the experiment results conducted on TPC-H benchmarked databases. Both the analysis and the experiment results show that the individualized watermark can withstand a variety of malicious attacks and benign updates, thus verify the feasibility and reliability of our proposed technique.

The other problem we are to address is *Invertible Watermarking* of precision-critical data. For the privileged users, the precision-critical data can be recovered without error, while being meaningless for non-privileged users. We introduce the idea of hierarchical access of the data, and enable the function of *Incremental Watermarking* when only a small portion of the database are modified or updated. Experimental results show that *Incremental Watermarking* is cost effective, making it suitable for real applications due to its low overhead.

1.3 Overview

The remainder of the thesis is structured as follow. The related works on digital watermarking and fingerprinting multimedia content, as well as watermarking relational databases are given in Chapter 2. Chapter 3 presents the *Individualized Watermarking* technique for relational databases. Chapter 4 covers the issue of *Invertible Watermarking* precision-critical databases. Experiment results for the above two techniques are included in Chapter 3 and Chapter 4 respectively. Finally, Chapter 5 gives conclusion and suggestions for future work.

Chapter 2

Literature Review

In this chapter, we shall discuss various approaches to watermarking relational databases in existing literature, and techniques of fingerprinting as well as invertible (lossless) watermarking multimedia content.

2.1 Watermarking of Databases

While work in the area of watermarking multimedia data and database security is proliferating, relatively less research has been done in watermarking relational databases. In a pioneering work of [2], Agrawal and Kiernan first raised the problem of database watermarking that marks the numeric attributes of relational data. A one-way hash function depending on a private key known only to the owner is used. The hash function algorithmically decides the tuples, attributes within a tuple, and bit positions (LSBs, the least significant bit) in an attribute to be marked, as well as computes the specific bit values after marking. Only if the attackers have access to the private key, can the watermark be detected with high probability. On the other hand, the attackers will either get caught for piracy or greatly reduce the database usability.

The strength of the work in [2] include:

- Identification of the rights management problem for relational data;
- Outline of some requirement difference between watermarking relational databases and multimedia data;
- List of a scope of possible attacks that the watermark is supposed to survive;
- Empirical evaluation of effectiveness and robustness of the proposed watermarking algorithm, etc.

Though the watermark survives several attacks and preserves mean and variance of the numeric data, yet it has no guarantee to survive distortion induced on queries. In other words, it does not fully consider the specific properties of relational databases.

In summary, [2] has the following weaknesses:

- The technique is a LSB watermarking depending on a probabilistic framework and an empirical detection threshold, so it does not preserve the integrity of the watermark sequence;
- It cannot preserve the key relationships of a database and the join constraints; also the uniqueness and relative constraints of the values are not carefully considered;
- The requirement that the LSB in any tuple can be altered may limit the applicability. For example, an important application such as data mining, requires the classification to be preserved.
- The vulnerabilities of LSB approach lie in its nature. [2] inherits all the disadvantages of LSB (least significant bits) watermarking techniques which has

been elaborated in [30]. For instance, If an attack assumes that the LSB space is insignificant, then simply altering the space randomly or zero-ing the LSBs would immediately defeat the watermark detection.

- Though it can prove the ownership of the database to some extent, it is not able to trace or prevent redistribution where there are multiple sources of piracy.

While [2] is the pioneering work in watermarking relational databases, it may not be a practical or useful solution for protecting real data. Also, a similar work reported in [1] has the similar problems.

Independently, [29] proposed algorithms on watermarking numeric datasets, which was later extended for watermarking relational data [30]. [30] pre-defines an order for the numeric dataset, and embeds the watermark bits into the data distribution properties instead of into the data themselves. For each selected subset S_i of the original dataset, a single watermark bit b is to be encoded into S_i. The watermark is modelled by the percentage of positive "confidence violators" present in S_i for a given confidence factor c and confidence violators hysteresis interval (V_{false}, V_{true}), where $V_{false} < V_{false}$, $c \in \{0, 1\}$ are real numbers (e.g. $c = 90\%, V_{true} = 10\%, V_{false} = 7\%$).

Some usability bounds metrics are defined to identify the range of acceptable changes to the data. Note that the acceptable level of changes is dependent on the intended application of the data.

One example of the usability metrics can be defined as *maximum allowable mean squared error*, where the mean squared error of each element of the data must be bounded by its corresponding allowable distortion, so is the sum of the mean squared errors over all the elements.

Aside from the allowable distortion on individual values and overall distortion limits, there are also some semantic features need to be considered. For example, it is critical that ages under 21 should remain so after watermarking if the attribute "age" is used to determine some behavioral patterns; result of classification should be preserved for some application; also, for a collection of relations, key relationship must be preserved, etc.

In the watermarking phase, if the bound is exceeded for some item, the current item is abandoned, and a new item for watermarking is chosen. The method embeds 1 bit per subset, so that there are a number of copies of the same bit over the databases.

In the detection phase, for each subset, the algorithm recovers all the watermark copies embedded in the data, which globally produces a set of copies of the same watermark bits with various errors. Then it uses a majority-voting over these recovered watermark bits to decide the most likely embedded bit. A trade-off is observed between the size of each subset and the bandwidth for watermarking. On the one hand, a larger subset size results in more correctly recovering the watermark bit, while lowering the watermarking capacity. On the other hand, a smaller subset size makes the watermark more fragile to attacks.

Comparing with [1, 2], the technique in [30] has the following advantages:

- The watermarking method is more resilient to attacks, because it encodes in the distribution domain of the data, instead of in the data themselves (e.g. LSBs).
- Unlike [1, 2], where the detection only determines the presence of the watermark, the detection phase of [29, 30] preserves the integrity of the watermark.

However, [30] has its own limitations:

- It is more costly in terms of the time complexity than [2] in that, for each attribute column, the mean need to be computed. And for each tuple to be watermarked, the corresponding attribute must be compared with the column mean. For another attribute of the same tuple, a new comparison is required.‘

- It does not address the issue of identifying different buyers to trace redistribution, either. Just as in [2, 1], the method proposed in [30] can not prevent the collusion attack in traitor tracing problems, etc.

In short, [1, 2] and [30] work well when there is one owner (or seller) of the relational data who would like to establish his/her ownership of the data. For this purpose, a watermark sequence is embedded in the numeric attributes. In case of an ownership dispute, i.e. some other person falsely claims ownership of the relational data, a neutral judge to whom the real owner can disclose the secret key can adjudicate the dispute. Then the judge will be able to detect the purported watermark. It is obvious that a fake owner will not be able to extract the embedded watermark. While the above techniques do solve the extremely important problem of establishing ownership, but did not address the problem of illegal redistribution. In practice, the valuable relational data can be sold to more than one buyer. If the cost of purchasing the data is high, a data pirate may be tempted to legally purchase one copy of the database (which may carry the owner's watermark) and then illegally redistribute copies of the data to other buyers at a drastically reduced price. Or alternatively, the pirate colludes with a few other malicious buyers to erase the watermarks. Even if the genuine owner suspects piracy, there is no way of tracing the precise leakage mechanism.

2.2 Fingerprinting of Databases

In [24], the watermarking technique in [1, 2] is generalized for fingerprinting databases. It uses a technique similar to that of [1, 2], but with a fingerprinting approach. Fingerprint bits are distributed uniformly across tuples, and the scheme is robust to a variety of attacks, including most of those mentioned in [1, 2], as well as collusion attacks.

However, just as in the case of [2], the technique requires that the relational tables have many numerical columns to be watermarked. In the case that only one or a small number of columns are available for watermarking, the accuracy and reliability of their technique may be greatly reduced. Most importantly, all the disadvantages of bit-plane watermarking elaborated hold for this technique. As pointed out in [30], LSB watermarking techniques have been tried and abandoned even in rights management of multimedia content.

Gross-Amblard [15] proposed the problem of watermarking databases or XML documents, while preserving a set of queries. Their approach works for a limited class of queries only. Moreover, it dose not consider the general problem of collusion attacks, where several buyers combine together to remove the watermarks. Notwithstanding, it briefly addressed the problem of *auto-collusion attack*, where a data owner needs to update the database and propagate changes to each of the registered data buyers, such that the same buyer can average the data from a few successive versions of the databases to remove the watermark. Therefore, they suggested a brute-force update on the databases instead of incrementally updating portions of the database when condition requires. However, there is neither security analysis nor experimental study in supporting their contribution on defeating *auto-collusion attack*. And clearly, it bears no traitor tracing ability at all.

Motivated by the shortcomings of the above work, we aim at developing a novel individualized watermarking algorithm which can prove ownership as well as identify the origins of unauthorized redistribution. Besides the attacks in the form of data re-sorting, subset selection, subset addition, etc, the individualized watermark should be buyer specific and resistant to collusion attacks.

In the light of the success of collusion-resistant watermarking for multimedia data, we would like to dig more into the literatures on the techniques catered to fingerprint multimedia content in the following section.

2.3 Collusion-resistant Watermarking

Watermarks (WM) and fingerprints (FP) are both widely used in digital rights management but they are utilized for distinct purposes. WM is used to designate the ownership of a dataset. Thus given the host data, the WM is identical in all the copies sold to establish ownership. On the other hand, FP is used to trace piracy so as to penalize redistribution. Therefore, FP should be individualized for various buyers of the data.

Several works have discussed the problem of fingerprinting for digital data(e.g., software, documents, and images). For example, Boneh and Shaw [6] proposed a collusion-secure fingerprinting method for digital data. They discussed methods for assigning codewords for the purpose of fingerprinting digital data. Fingerprinting consists of uniquely marking and registering each copy of the data. This marking allows a distributor to detect any unauthorized copy and trace it back to the specific buyer.

In [11], distinct spread-spectrum watermark sequences were embedded for different copies of the data, and collusion attacks were modelled as averaging of copies with additive noise.

Fiat et al [12] introduced a dynamic traitor-tracing scheme where users are randomly grouped into r subsets, each receiving a distinct symbol. And the tracing is confined in the subset that includes the pirate only.

[22] presents a novel collusion-resistant watermarking technique that can achieve a minimum collusion size that grows linearly with the number of copies of the individ-

ually marked data. Using a secret key, the various copies of the multimedia content can be identically watermarked, but the individual detection keys are distinct [22]. Knowing one detection key, the attackers cannot remove the watermark or retrieve a watermark-free copy of the data without making the data useless. Also if the watermark in one copy is destroyed, enough information about the broken detector key (fingerprint) can be known. Thus, the technique in [22] can identify those who participated in a collusion attempt and has traitor-tracing ability.

Given the watermarking key w (referring to the notations in [22]), the i^{th} watermark detection key h_i is defined as $h_i = w + c_i$, where c_i denote the watermark carrier, which is different for each buyer and hence it can be considered to be a fingerprint which identifies each buyer. Obviously, h_i is different from w and is distinct for different buyers of the data. The security of the scheme lies in the fact that even if the attackers break the detection key h_i, as h_i does not deterministically reveal w, they still cannot remove w from the watermarked multimedia content.

For watermark detection, the detector can similarly do a correlation test on the watermarked data y_i of the i^{th} buyer and the watermark sequence w by using the classical Neyman-Pearson hypothesis test $d_W = y_i{\cdot}w$ to decide whether the watermark is present or not. It decides that the watermark is present and thus proves the ownership if $d_W > \delta_W$, where d_W is the detection threshold that controls the tradeoff between the probabilities of false positive and false negative decisions. Otherwise, the watermark is considered not to be present. Similarly, another test can be done to reveal the presence of the i^{th} buyer's fingerprint. $d_H = y_i \cdot h_i$. If $d_H > \delta_H$, the fingerprint of the i^{th} user is detected. Otherwise, the fingerprint of the i^{th} user is not present.

However, the severe limitation of [22] is that if the attackers do not break the detection keys, and simply distribute the copies without removing the watermark information, the seller can only prove ownership of the distributed copies, but cannot detect the actual illegal distributors. This is not really a limitation under the assumptions in [22] because they address the problem of multimedia content distribution which assumes that the content has to be played on some standard playback device which does this type of detection. In that scenario, since each playback device has an individualized detection key, the above limitation does not really arise since no one can make use of the data without the playback device. This, however, shows that the method in [22] is not applicable for watermarking relational data since one cannot assume the mandatory use of only tamper-proof devices for accessing the data.

In summary, while the current database watermarking mechanism does not have traitor tracing ability and does not survive collusion attacks, adaptation of the existing methods of collusion-resistant watermarking multimedia data to watermark databases is non-trivial.

While our work in Chapter 3 is inspired by the rigorous approach in [22], it is quite distinct from their work, given the distinct requirements of watermarking relational databases from watermarking multimedia data. It can both establish ownership and precisely and reliably identify colluders, no matter the attackers attempt to remove the watermark or not.

2.4 Invertible Watermarking Mechanisms

Invertible watermarking is also referred to as reversible watermarking, lossless watermarking, and distortion-free watermarking in literature.

In [14], an invertible watermarking for JPEG image authentication is presented. The assumption that original images have been lossy-compressed is a drawback. Even after the watermark is removed, the result is still lossy-compressed image. In [13], Fridrich et al improved the idea and applied it to watermark all image formats.

A most recent work in [9] presents a spread-spectrum invertible watermarking system for authenticating images in lossless formate. The integrity of the images can be verified and original images before embedding watermark can be recovered.

However, the techniques in both [9] and [10] are not directly applicable to watermark relational databases, given the difference between multimedia data and relational databases described in Chapter 1.

2.5 Summary

In this chapter, we reviewed the related literatures of our work. We first introduced some works on watermarking of databases and on fingerprinting of databases. Due to the weaknesses in these techniques, we moved on to study the works done on collusion-resistant watermarking of multimedia data. Also, we discussed the techniques on invertible watermarking of images. Chapter 2 paved way for solving the problems focused on in the following chapters.

Chapter 3

Individualized Watermarking of Databases

In this chapter, we consider the more general buyer-seller scenario where an owner of the database sells it to many different customers. For this case, some new difficult requirements arise. First, there is the need to individually identify each copy of the data sold in order to precisely and reliably identify the source in case of piracy. Most of the earlier techniques are not able to distinguish the various buyers of the same data. Thus, some form of individualized watermark for each buyer is needed for every copy sold. Secondly, such a scheme would be susceptible to collusion attacks in which a set of legitimate buyers collude together to illegally tamper the watermark. We present a novel watermarking algorithm based on the direct sequence spread-spectrum technique which embeds the owner's watermark as well as the individual buyer's fingerprint into each copy of the relational data. Our security analysis shows the feasibility of the proposed technique for real applications. We have applied this watermarking technique to relational database generated based on TPC-H Benchmark. Experiments conducted on a commercial database system confirm that the proposed method can survive attacks such as data re-sorting, subset selection, subset addition etc., as well as collusion attacks.

Our approach has four novel features: (1) it is a direct sequence spread spectrum based watermarking technique that generates a pseudo-random fingerprint for each buyer; (2) it is robust against various attacks and has low distortion introduced to the data values without compromising the integrity (mean and variance) of the data; (3) it makes use of a hash table to define and later restore the order of tuples, so that the deleted or inserted tuples will not affect the performance of watermark detection; (4) the correctness (with low numeric distortion within tolerance)of queries (e.g. join and project) on the database is preserved. Our proposed individualized watermarking technique is buyer specific and thus resistant to collusion attacks. We present a security analysis on this and show that the colluders will fail in effectively removing the watermark. Furthermore, we have implemented our individualized watermarking technique on a widely used database management system MySQL [25], and have experimentally verified the robustness of our scheme using the TPC-H benchmark data. Attacks in the form of data re-sorting, subset selection, subset addition, etc. as well as collusion attacks were conducted, and none have caused problems to the proposed method.

This chapter is structured as follows. Section 3.1 outlines the major attacks that a watermarking database scheme is subject to. Section 3.2 introduces the fundamentals of spread-spectrum and collusion-resistant watermarking. Section 3.3 refines the algorithms in Section 3.2 in order to adapt it to watermark relational databases, which overcomes the limitations of the previous approaches. Section 3.4 provides a detailed analysis of the security of our scheme, including the robustness against common database ownership attacks and collusion attacks. Section 3.5 presents experimental results. Finally, Section 3.6 summarizes the chapter.

3.1 Possible Attacks

Given the peculiarities of relational databases, we first systematically outline the possible attacks that databases watermarking may suffer. Note that an attack is successful only if it preserves the usability of the data. The following attacks come either in the form of benign updates by regular database update operations or the form of malicious attacks aiming at destroying the watermark or claiming false ownership.

A1. Bit Modification: The attacker can simply update the least or least few significant bit(s) of the entire data set. Though the attack is simple, the effect is devastating, especially to LSB watermarking. The attack can be done by flipping all the LSBs or setting all LSBs to '0' (or '1').

The work in [2] suffers exactly from this Attack **A1**, although the exact number of LSBs watermarked is not known to the attackers.

A2. Tuple Reshuffle: The attacker reshuffles the relational database tuples. Thus, the re-sorted data set loses synchronization with respect to the watermark detection key, which increases the difficulty of watermark detection.

A3. Subset Selection: The attacker randomly selects or deletes a subset of the original dataset tuples. Thus, some of the watermark bits are lost.

This attack can be understood from two opposing directions. On the one hand, the attacker arbitrarily deletes some tuples in the hope that the mark is removed from the remaining part. On the other hand, the attacker selects randomly a set of tuples and wishes the selected set would not contain the mark. Interestingly, the attack seems rendering the defense conflicting. Intuitively, the more bits we insert, the more possible some of inserted bits are removed in the first case; in contrary, the less bits we insert, the less possible the inserted bits exist in the second case.

Basically, A3 suggests the relative size of the set of watermarked tuples not to be small. Under this rationale, a sound solution unifying the above two plausibly conflicting scenarios of defense is as the following: a mark is repeatedly inserted many times so that a large portion of the data is watermarked.

A4. Subset Addition: The attacker adds a number of new tuples to the dataset. This can also render the data losing synchronization in the watermark detection phase.

A5. Attribute Projection: The attacker may project a subset of attributes that are useful to him and create a new relational table. Thus, a number of columns are missing and part of the watermark is destroyed after this attack.

A6. Attribute Permutation: This attack is quite simple in that the attacker does not need to make any change to the data values, but to simply permutate the attributes. *Attribute Renaming* is assumed as a special case of this attack. Similar to Attack **A2.**, the watermarked data lose synchronization with respect to the watermark detection key, which renders the watermark detection more difficult.

This attack suggests that the names or sequence numbers of the attributes should not be relied on in the detection phase. So a sound watermarking scheme should base the recognition of an attribute on some intrinsic properties, e.g., the mean and variance of a numeric attribute.

A7. Collusion Attack: Typically, a collusion attack has two forms:

(1) Some attackers work together to remove the watermark, so that the resulting data are uncorrelated with the detection keys of any of these attackers.

(2) Some attackers average several copies of watermarked data, so that the resulting data are not correlated with the detection keys of any of these attackers.

Attack **A7** is more challenging than the others. There have been some proposed solutions to this problem for multimedia data [6, 22, 10], however, the feasibility of direct application of them to the case of watermarking relational data deserves future efforts.

3.2 Spread-Spectrum Watermarking

While our work here is inspired by the rigorous approach in [22], it is quite distinct from their work, given that their scheme works only for multimedia content, and cannot be adapted to watermark relational databases easily. Before we can describe our individualized watermarking technique, we first give a brief introduction of the traditional spread-spectrum mechanism.

In spread spectrum communications, a narrow-band signal is transmitted over a much larger bandwidth such that the energy present in any single frequency is imperceptible [8]. It is a technique whereby an already modulated signal is modulated a second time, in such a way that it produces a signal which interfaces in a barely noticeable way with any other signal operating in the same frequency band. The interfering signals are transparent to the spread spectrum signals, and the spread spectrum signals are transparent to the interfering signals. To achieve transparency, the spread spectrum modulation decreases the transmitted power spectral density so that it lies well below the thermal noise level of any unfriendly receiver.

When the spread spectrum technique is applied to watermarking, the watermark is spread over very many frequency bins (or relational data tuples in our approach) so that the distributed energy (distortion) in any frequency bin (tuple) is very small and thus undetectable. The watermark is therefore like a pseudo random noise which is below the tolerance threshold and thus cannot be identified. Nevertheless, because the watermark verification process knows of the precise location and the content of the watermark, it is possible to concentrate these weak signals into a single signal with a high signal-to-noise ratio. However, to destroy such a watermark would require noise of high amplitude to be added to all frequency bins (tuples in our case), as any attacker who attempts to confidently eliminate the watermark, must introduce much greater distortion to the data than the distortion from watermarking. As a result, an attack creates obvious defects in the data.

Three schemes are commonly used for spreading in radio-frequency communications: direct sequence, frequency hopping, and chirp [8]. Since there is no frequency domain transformation for relational data, only the direct sequence spread spectrum (DSSS) technique is applicable here. In the DSSS scheme (which is most commonly used for watermarking), the signal is modulated by a function that alternates pseudo-randomly between $+\alpha$ and $-\alpha$, a gain factor, at multiples of a time constant called the chip rate. For relational data watermarking applications, the chip rate is the number of times a watermark bit is replicated. The bit is modulated by a pseudo-random carrier that contains components of all frequencies. It leads to spreading the modulated signal's energy over a large frequency band, rendering it undetectable at any individual frequency (tuple). Therefore, at any frequency (tuple), the distortion is quite low. We will now mathematically establish this process.

Let a_j, where $a_j \in \{-1, 1\}, j = 0, 1, 2, ..., M_0$, be a sequence of bits, which is then spread by a large factor cr, called the chip rate or the spread factor, to obtain the spread sequence b_i. The cr and M_0 are selected in such a way that $cr \times M_0 = N$, where N is the size of the host data z.

$$\forall j : \;\; b_i = a_j, \;\; j \cdot cr \leq i < (j+1) \cdot cr.$$

The spread sequence b_i is then amplified with an amplitude factor α and modulated with a binary pseudo-noise sequence r_i, where $r_i \in \{-1, 1\}$, $i = 1, 2, ..., N$, yielding the modulated signal, i.e. the watermark

$$u_i = \alpha \cdot b_i \cdot r_i.$$

Given the host data z_i, $i = 0, 1, 2, ..., N$, the watermarked data z'_i is computed as follows:

$$z'_i = z_i + u_i = z_i + \alpha \cdot b_i \cdot r_i, \;\; i = 1, 2, ..., N.$$

Due to the noisy nature of r_i, u_i is also a noise-like signal and thus difficult to detect, locate and manipulate. Note that the amplitude factor α can be thought of as the tolerance of a particular numeric attribute. The recovery of the hidden information is easily accomplished by correlating the watermarked signal with the same pseudo-noise sequence r_i that was used in the embedding phase:

$$S_j = \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i \cdot z'_i = \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i \cdot z_i + \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i^2 \cdot \alpha \cdot b_i.$$

The first term on the right-hand side of the above equation vanishes if

$$\sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i \cdot z_i = 0.$$

Note that the pseudo-noise sequence $\{r_i,\; j \cdot cr \leq i < (j+1) \cdot cr - 1\}$ contains as many -1's as 1's, and r_i and z_i are uncorrelated. Theoretically, $\sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i \cdot z_i = 0$. In practice however, this sum is not zero such that a correction term

$$\triangle = -(\sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i) \cdot mean(z_i),$$

which accounts for the different number of -1's and 1's in the pseudo-noise sequence, has to be added. S_j then ideally becomes

$$S_j = \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i \cdot z'_i + \triangle + \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i^2 \cdot \alpha \cdot b_i \approx \;\; 0 + \sum_{i=j \cdot cr}^{(j+1)\cdot cr-1} r_i^2 \cdot \alpha \cdot b_i = cr \cdot \alpha \cdot a_j.$$

Since α and cr are positive, we have

$$sign(S_j) = sign(cr \cdot \alpha \cdot a_j) = a_j.$$

Therefore, the hidden information a_j can be recovered as

$$a_j = sign(S_j).$$

The above correlation is useful for watermark bits extraction. In the latter part of this chapter, we will discuss how the technique can be used in watermark detection.

3.3 Our Approach

In this Section, we propose our individualized watermarking for Relation Databases. Notations used in our algorithms and the analysis are summarized in Table 3.1.

3.3.1 Sorting the dataset

In image watermarking, pixels have fixed relative positions. Unfortunately, the tuples of a relational database have no defined order. However, a fixed order is necessary to detect the embedded watermark and keep the detection phase synchronized with the embedding phase. Thus, some invariant information in databases can help define such an order. For our technique, we choose the primary key information for this purpose because the primary key information usually cannot be changed or removed. Otherwise, the database becomes significantly less useful. This is also the assumption of the work in [2]. Therefore, we assume that it is highly unlikely that the attackers will intentionally change or remove the primary key information of databases, as it

Notation	**Description**
t	a tuple with schema $R(P, A_1, A_2, ..., A_{\nu-1})$
P	the primary key
sk	a secret key
x	the host data
N	size of the host data
A	standard deviation of x
w	watermark sequence to establish ownership
L	size of w
M	number of buyers
p_i	pseudo-random binary sequence for the i^{th} buyer
c_i	fingerprint for the i^{th} buyer
B	standard deviation of c_i
w_i	individualized watermark for the i_{th} buyer
cr	the chip rate
h_i	spread version of w_i, the detection key
τ	tolerance of the host data
y_i	watermarked data for the i^{th} buyer
$\widehat{y_i}$	possibly modified version of y_i
d_W	correlation value
g_W	detection noise
δ_W	individualized watermark detection threshold
v^*	estimation attack vector
K	size of collusion clique
ε_1	probability of false positives
ε_2	probability of false negatives

Table 3.1: Notations used in Chapter 3

reduces the utility of the database information.

We assume that a standard cryptographic hash function like SHA-1 or MD-5 can be used to establish order [31, 2]. Given a database relation R with schema $R(P, A_1, A_2, ..., A_{\nu-1})$, where P is the primary key, the hash value $F(t \cdot P)$ of primary key attribute P of tuple t can be computed using:

$$F(t.P) = H(sk \circ H(sk \circ t.P)),$$

where sk is a secret key and H is a cryptographic hash function like SHA-1 or MD-5, and $\circ$ denotes concatenation. The hash values can then help define the order of the tuples of the relational data. We store the hash values in a table, which should be maintained for the future watermark detection need.

There is a primary key replacement attack, such that the attacker simply hashes the primary key and makes use of the hash value to replace the primary key. In the way, the primary key is still unique for each tuple, which makes the database suitable for some application. All the existing techniques are subject to this attack. However, as we previously assumed, the modification of primary key will render the data significantly less useful. Especially when the primary keys carries meanings instead of merely serving as ID's, this attack will make the primary keys totally meaningless. Moreover, when join operations are applied on two or more tables, improper modification of primary keys or foreign keys will introduce great errors to the join results. In short, attackers can hardly benefit from this primary key replacement attack.

3.3.2 Embedding Phase

Like [2], our technique watermarks and fingerprints only the numeric attributes. Without loss of generality, assume the first k numeric attributes $A_1, A_2, ..., A_k$ are candidate of watermarking and fingerprinting. Figure 3.1 outlines the individualized watermark embedding algorithm.

Let w be a sequence of binary values $\{-1, 1\}$ of size L. w is for the purpose of establishing ownership, and it is the same for all the buyers. We assume that one buyer corresponds to one copy of relational data sold. Let $C = \{c_{ij}\}$ be a $M \times L$ matrix, where M is the number of buyers, and $c_{ij} \in R, c_{ij} = N(0, B^2)$, i.e. each entry is a zero-mean normal random variable with standard deviation $\sigma_c = B$. Let $P = \{p_{ij}\}$ be a $M \times L$ matrix of pseudo-random binary values, where $p_{ij} \in \{-1, 1\}$.

For each buyer i, p_i is distinct for this buyer and is uncorrelated with any other buyer's sequence p_j, $\forall j : i \neq j$. To guarantee this, good pseudo-random sequences possessing certain properties in terms of sequence length, auto-correlation, cross-correlation, orthogonality, and bit balancing are recommended. For example, good pseudo-random sequences like M-sequence and Gold sequence of length 31 [28] can be used.

The individualized watermark sequence w_i (here p_i, c_i and w_i all denote sequences, instead of bits)can be computed as:

$$w_i = p_i \cdot w + c_i.$$

Note that the goal of the watermark carrier c_i is to hide the watermark w, and the goal of p_i is to further de-correlate the different w_i's.

Then we spread the individualized watermark sequence w_i by the chip rate cr, such that the spread watermark h_i is of size $N = cr \cdot L$, where N is the size of the host data x (i.e. the sequence of numeric attributes selected to be watermarked). Each window of w_i corresponds to one watermarking block. In total, we have cr blocks.

We then embed the spread individualized watermark h_i into the host data x, where x is relational data sorted by the above sorting algorithm, to obtain the watermarked data y_i for the i^{th} buyer.

$$y_i = x + \frac{\tau}{2} h_i,$$

where τ denotes the tolerance of the host data. We assume τ is the maximum distortion that the numeric attribute of the tuple can survive without compromising the integrity of the data. Note that for different relational data, τ can be different. Even for the different columns in the same table, τ can be varied as well. Note that up to cr number of copies of the individualized watermark w_i are embedded into the relational data. Since we choose factor cr as large as possible, we maximize the spreading of the embedded watermark bits w_i. This results in the advantages of the spread spectrum technique which increases robustness (against various attacks) with low distortion introduced into the host data.

Individualized Watermark Embedding Algorithm:

```
Input: sorted relation $R$ with schema $R(P, A_1, A_2, ..., A_{\nu-1})$, number of tuples $N$, ownership sequence $w$ and its length $L$, $i^{th}$ pseudo random sequence $c_i$ and $p_i$
Output: new relation $R'$ with individualized watermark, chip rate $cr$
Method:
1: compute the individualized watermark $w_i$ for user $i$, $w_i = p_i w + c_i$
2: chip rate $cr = floor(N/L)$
3: $\forall\, m : \ m\ (mod\ L) = n$,   set   $h_{im} = w_{in}$
4: for each $j \leftarrow 1$ to $N$ do
5:     for each $l \leftarrow 1$ to $k$ do
6:         compute the watermarked data $A'_{j\,l} = A_{j\,l} + \frac{\tau}{2}\, h_{ij}$
7: return $R' = (P, A'_1, A'_2, ..., A'_k, A_{k+1}, ..., A_{\nu-1})$,  $cr$
```

Figure 3.1: Individualized Watermark Embedding Algorithm

3.3.3 Detection Phase

We now describe the watermark detection process. Note that we assume the scenario wherein several copies of the relational data are sold to different buyers, each copy embedded with a distinct buyer fingerprint c_i and the ownership watermark w modulated by p_i. In case of piracy or ownership dispute, the watermark and fingerprint need to be detected in a possibly modified copy of the watermarked relational data.

Figure 3.2 outlines the individualized watermark detection algorithm. The algorithm consists of 2 phases. Phase 1 re-sorts the tuples of a suspected copy of relational data according to the hash function and kept hash table. Phase 2 scans all the tuples in re-sorted order, and computes the correlation value between the suspected data $\widehat{y}_i$ and the corresponding detection key w_i to determine traitor(s). In the following, we describe the two phases in detail.

Semi-blind Re-sorting Scheme

Note that the suspected copy of the relational data can be a modified version $\widehat{y}_i$ of the original watermarked data y_i, which may be caused by distortion due to benign updates or malicious attacks. The primary key of each tuple is hashed by the same hash function used in the watermark embedding phase, and the hash value is compared with the kept hash table to check if the current tuple is present in the table. If it is present, the tuple is recovered in the order that the hash function indicates. Otherwise, the current tuple is a newly-inserted one which does not exist in our original table, so we simply skip and ignore it. After checking all the tuples in $\widehat{y}_i$, all the tuples that were present in the original database, are sorted and ready for detection purpose.

Since we use the hash table to recover the original order and ignore the newly-

Individualized Watermark Detection Algorithm:

Input: Hash secret key sk, database relation R' with schema $R'(P, A'_1, A'_2, ...A'_k, A_{k+1}, ..., A_{\nu-1})$, chip rate cr, suspected buyer ID l and corresponding detection key w_l

Output: buyer ID l if the l^{th} buyer pirates

Method:

phase 1: database re-sorting

```
1: for each tuple $t \in R'$ do
2:   compute $F(t.P) = H(sk \circ H(sk \circ t.P))$
3:   scan the kept Hash table
4:   if $F(t.P)$ is in the table then
5:     restore its order by $F(t.P)$
6:     retrieve the corresponding watermarking bit in $w_i$
7:     mark $isnew = 0$
8:   mark $isnew = 1$   /* this tuple is new*/
```

phase 2: scan all tuples and compute correlation

```
1: initiate the counts $count_+ = 0$
2: for each tuple $t$ in the re-sorted order do
3:   for each watermarking block $i \leftarrow 0$ to $cr - 1$ do
4:     for each tuple $j \leftarrow L \times i + 1$ to $L \times (i + 1)$ do
5:       if $isnew = 0$ then
6:         current tuple is involved in the calculation of correlation $d_W$ with detection
           key bit $w_{i[j\,(mod\,L)]}$
7:       else
8:         skip current tuple
9:     if $d_W > \delta_W$ then
10:      $count_+ = count_+ + 1$
11: if $count_+ > \frac{1}{2} \cdot cr$ then
12:   return traitor ID $l$
13: else
14:   return 0
```

Figure 3.2: Individualized Watermark Detection Algorithm

inserted tuples, it is not necessary to compare the current database with the original one. However, we have to maintain the hash values of the original tuples for re-sorting purpose. Therefore, our scheme can be considered as a *semi-blind* watermarking detection scheme [7] (differing from absolutely *blind* watermarking scheme, where no information from the original data needs to be maintained), which makes our scheme more competent for real database applications.

Correlation-based Watermark Detection

Given the sorted watermarked data $\widehat{y}_i$ (possibly attacked), we can compute the correlation of the watermarked data with the watermark detection key $w_i = p_i w + c_i$. Note that we compute the correlation for each watermarking block. For each block, we compute $d_W = \widehat{y}_i \cdot w_i$ and compare it with the detection threshold δ_W.

If the corresponding tuple of a watermark bit has been deleted, we ignore the tuple and the corresponding bit in w_i. Therefore, the number of bits in w_i involved in the correlation computation may be less than or equal to the actual length of w_i, due to the possible delete operations in the database. If the correlation value is greater than the detection threshold, i.e. $d_W > \delta_W$, then the watermark $w_i = p_i w + c_i$ is considered to be present. Otherwise, the watermark does not exist in that copy of the relational data.

Since we embed individualized watermark in all the k numeric attributes $A_1, A_2, ..., A_k$, we can actually compute k correlation values $\{d_{W_n}, n = 1, 2, ..., k\}$ for each watermarking block. Thus, we can choose the maximum value in $\{d_{W_n}, n = 1, 2, ..., k\}$ as the value of d_W, i.e.

$$d_W = max(d_{W_n}), \forall n : n = 1, 2, .., k.$$

Also, as we embed the individualized watermark $w_i = p_i w + c_i$ up to cr times in the host data, we can compute cr number of d_W for all the watermarking blocks. Then we can apply a majority voting scheme to decide whether the watermark is present or not.

In the event that collusion attack occurs, we apply phase 2 of the detection algorithm to all the suspected buyers' detection keys. Thus, we can detect the traitors (malicious buyers) who took part in the collusion attack.

The major advantage of our scheme is that we can establish the ownership and detect the real colluders, even if the attackers collude and tamper the watermark information. In the following section, we will present the formal security analysis.

3.4 Security Analysis

If there are no malicious attacks or benign updates, we denote the watermarked relational data for the i^{th} buyer as:

$$y_i = x + \frac{\tau}{2}(p_i w + c_i),$$

where τ denotes the tolerance which is the amount of distortion that the content can survive. Note that τ can vary according to different relational data and different numeric attributes. Let $w_i = p_i w + c_i$ denote the corresponding detection key for y_i.

We denote correlation between y_i and w_i as $d_W = y_i \cdot w_i$, which is the normalized inner product of vectors y_i and w_i. For instance,

$$a \cdot b = \frac{1}{\| a \| \| b \|} \sum a_i b_i,$$

with $a^2 = a \cdot a$.

Let δ_W be the watermark detection threshold, which decides that the watermark is present if $d_W > \delta_W$. Basically, δ_W controls the trade-off between the probabilities of false positives and false negatives, i.e. for equal probability of false positives and false negatives, we should set $\delta_W = 1/2$ (refer to [22]).

Let v_i be the attacker's estimated version of w_i. The closer the estimated v_i is to w_i , the more successful is the attacker in eliminating the watermark.

For watermark detection, let ε_1 denote the probability of false positives, which is the probability of wrongly identifying an unwatermarked copy of the data as watermarked. For the sake of the credibility and security of our scheme, ε_1 must be kept very small with a typical value of $\varepsilon_1 = 10^{-9}$.

Let ε_2 denote the probability of false negatives, which is the probability of not identifying a colluder. We would like ε_2 to be small as well, but do not have to insist that ε_2 is as small as ε_1.

3.4.1 Watermark Removal Attack

Assume the pseudo-random number generator is secure (i.e. PRNG can not be attacked by attackers), let us consider the case where the attacker has broken the watermark detection key $w_l = p_l w + c_l$ for client l. Since the attacker does not know which copy $y_l = x + \frac{\tau}{2}(p_l w + c_l)$ the detection key w_l corresponds to, he/she is likely to utilize w_l to estimate the watermark $w_i = p_i w + c_i$ for the buyer i, where it is with high probability that $l \neq i$. The attacker estimates the detection key $v_i = \alpha w_l = \alpha(p_l w + c_l)$, where $\alpha \in R$. The attacker removes v_i and creates a new version of relational data content by:

$$\widehat{y}_i = y_i - v_i = y_i - \alpha w_l = x + \frac{\tau}{2}(p_i w + c_i) - \alpha(p_l w + c_l).$$

In order to deceive the detector that the watermark is not present in the attacker's modified data, the attacker would like to lower the correlation value given by $d_W = \widehat{y} \cdot w_i$:

$$d_W = \widehat{y}_i \cdot w_i = [x + \frac{\tau}{2}(p_i w + c_i) - \alpha(p_l w + c_l)] \cdot (p_i w + c_i).$$

Note that we assume the fingerprint part c_i follows a zero-mean Gaussian probability distribution with the standard deviation $\sigma = B$. For the sake of simplicity of analysis, we set $\tau = 2$.

If somehow, by sheer chance, the attacker has the knowledge of $\widehat{y}_l = x+p_lw+c_l$ and corresponding w_l, he can simply remove the watermark, i.e $\widehat{y}_l = x+p_lw+c_l-\alpha(p_lw+c_l)$ to drive $E[d_W] = 0$. In this case, it is obvious that the attacker need to set $\alpha = 1$. Therefore, even if the attacker does not know $\widehat{y}_l = x + \frac{\tau}{2}(p_lw + c_l)$, he will set $\alpha = 1$ as well.

Given $\alpha = 1$, the correlation value becomes:

$$d_W = \widehat{y}_i \cdot w_i = [x + \frac{\tau}{2}(p_iw + c_i) - (p_lw + c_l)] \cdot (p_iw + c_i),$$

and its expect value can be computed as $E[d_W] = 1 + B^2$, which is already large enough for us to detect the existence of w_i.

Thus, it is apparent that even with the knowledge of $h_l = p_lw + c_l$, the attacker will not be able to remove watermark w_i from some copy of the watermarked data y_i, where $i \neq l$.

In essence, the attacker cannot succeed in the watermark removal attack.

3.4.2 The Collusion Attack

Consider a collusion clique of size K. Assume that the attacker(s) have broken K detection keys $\{w_i = p_iw + c_i,\ i = 1, 2, ..., K\}$, but do not know how these detection keys correspond to the watermarked data $\{y_i,\ i = 1, 2, ..., M, M >> K\}$. Or in another type of attack, K legitimate buyers collude together to defeat our watermark detection. Since the attackers will either fail in removing the individualized watermark or make the data useless, we assume that the attackers' real goal is to not to totally remove the watermark, but to make the attackers' modified version $\widehat{y}_i = y_i - v_i$

uncorrelated with any of the colluders' detection keys $\{w_i = p_i w + c_i,\ i = 1, 2, ..., K\}$, so that the colluders cannot be caught.

Watermark Estimation Attack

With the knowledge of K detection keys $\{w_i, i = 1, 2, ..., K\}$, the attacker would like to best estimate the attack vector v^*, in a manner such that $\widehat{y}_i = y_i - v^*$ does not show significant correlation with any of watermark detection keys $\{w_i\}$, even for $i > K$.

We will now show that even with the knowledge of K detection keys $\{w_i, i = 1, 2, ..., K\}$, it is still difficult for the attackers to estimate a *good* attack vector v^*. We name the use of the estimate v^* as **Watermark Estimation Attack**.

Lemma 3.4.1 *The estimation attack vector v^* is given by*

$$v^* = \frac{1}{K}(\sum_{i=1}^{K} w_i)$$

Proof: *Watermark estimation attack aims to find a vector v^* that minimizes the average distance between the vector and the actual watermark detection key w_i, i.e. $dist_i = \frac{1}{K}\sqrt{\sum_{i=1}^{K}(w_i - v^*)^2}$. We therefore have*

$$dist_i{}^2 = \frac{1}{K^2}(w_i - v^*)^2 = \frac{1}{K^2}(\sum_{i=1}^{K} w_i{}^2 - 2(\sum_{i=1}^{K} w_i) \cdot v^* + K \cdot v^{*\,2}).$$

To minimize all the distances $\{dist_i,\ i = 1, 2, ..., K\}$, it is obvious that we should set $v^* = \frac{1}{K}(\sum_{i=1}^{K} w_i)$. [QED]

Collusion Size to Defeat Watermark Detection

Given the concept of watermark estimation attack, we now prove that the attackers (or colluders) will fail in effectively removing the watermark and thereby defeating the watermark detection.

Lemma 3.4.2 *Given K detection keys $\{w_i = p_i w + c_i, i = 1, 2, ..., K\}$, in order to reduce the correlation value to $E[d_W] < \theta$, where $\theta < 1 + B^2$, Condition 1 must hold.*

Condition 1: *the K keys must contain the detection key w_l for the attacker's copy $y_l = x + w_l$.*

Proof: *First, assume that w_l is not in the K detection keys $\{w_i = p_i w + c_i, i = 1, 2, ..., K\}$. By using the watermark estimation attack, the attacker's version is*

$$\widehat{y}_l = x + p_l w + c_l - \frac{1}{K}\sum_{i=1}^{K} p_i w - \frac{1}{K}\sum_{i=1}^{K} c_i.$$

With $d_W = \widehat{y}_l \cdot w_l = (x + p_l w + c_l - \frac{1}{K}\sum_{i=1}^{K} p_i w - \frac{1}{K}\sum_{i=1}^{K} c_i)(p_l w + c_l)$ and $l \neq i,\ \ i = 1, 2, ..., K$, we can compute $E[d_W] = 1 + B^2 > \theta$, so that the attackers will be caught in piracy and collusion. Therefore, Condition 1 must be satisfied in order for the attack to succeed. [QED]

To ensure Condition 1, it is obvious that a relatively large K is desired. However, the larger K is, the more likely it is that the collusion clique will be betrayed by one of the colluders. Also, it is more difficult and expensive to make a collusion clique of a larger size.

As a result of the above contradiction, the attackers cannot succeed in removing watermark by collusion attack.

Averaging Collusion Attack

Lemma 3.4.3 *Given K different buyers' watermarked relational data, in order to reduce the correlation value to $E[d_W] < \theta$ after the averaging attack, the collusion size should be $K > \frac{1+B^2}{\theta}$.*

Proof: *Assume that attackers average K different buyers' data to obtain a new version of the relational data $\widehat{y}_{new} = x + \frac{1}{K}\sum_{i=1}^{K} p_i w + \frac{1}{K}\sum_{i=1}^{K} c_i$. We would like to use the detection keys $\{w_i = p_i w + c_i,\ \ i = 1, 2, ..., K\}$ to detect the existence of the watermark, i.e.*

$$d_W = \widehat{y}_{new} \cdot w_i = (x + \frac{1}{K}\sum_{i=1}^{K} p_i w + \frac{1}{K}\sum_{i=1}^{K} c_i)(p_i w + c_i).$$

In order to keep $\forall i :\ \ E[d_W] = \frac{1}{K}(1 + B^2) < \theta$, it is obvious that $K > \frac{1+B^2}{\theta}$. [QED]

Clearly, the smaller the value of θ is, the larger is the collusion size K required. For a given θ value, the minimum collusion size grows proportional to B^2. For example, if $\theta = 0.2$, then $B^2 = 0.8$, the attackers need to average $K > 9$ copies. For $\theta = 0.2$, $B^2 = 2$, the attackers have to average $K > 15$ copies.

Also note that the larger the collusion size K is, the more likely it is that the collusion clique will be betrayed by one of the colluders. So usually the attackers aim to keep the clique size as small as possible, e.g. $K \leq 3$. These above two facts contradict each other. Therefore, our scheme is resistant to such a collusion attack.

Thus, from Lemma 3.4.2 and Lemma 3.4.3, we can conclude that the collusion attack is highly unlikely to succeed.

3.4.3 Probability of False Positives

In this section, we would like to estimate the probability of falsely detecting a watermark when there does not exist any. For this purpose, we compute the detection noise g_W, which is defined as $d_W = 1 + g_W$ if $\widehat{y}_i$ was marked; and $d_W = 0 + g_W$, otherwise. We assume g_W to have a zero-mean Gaussian distribution. If there are no attacks, then

$$d_W = y_i \cdot w_i = (x + p_i w + c_i) \cdot (p_i w + c_i) = 1 + g_W.$$

Therefore, $g_W = x \cdot p_i \cdot w + x \cdot c_i + 2p_i \cdot w \cdot c_i + {c_i}^2$. Since g_W has a normal distribution, it can be shown that the detection noise variance is

$${\sigma_{g_W}}^2 = (A^2 + 2B^2 + 5A^2B^2 + 5B^4)/N$$

where A is the standard deviation of the host data, and N is the size of the host data. For a given value of σ_{g_W}, we can now calculate the probability of false positives $\varepsilon_1 = Pr[\, d_W > g_W \mid \textit{dataset is not marked}\,]$ as follows:

Corollary 3.4.1 *A dataset of size N produces*

$$\varepsilon_1 = \frac{1}{2} erfc(\frac{\delta_W \sqrt{N}}{\sqrt{2(A^2 + 2B^2 + 5A^2B^2 + 5B^4)}}),$$

where $erfc(\cdot)$ is the complementary error function [22].

From Corollary 3.4.1, we can observe that to achieve a given ε_1, with the increase of N, we should lower the detection threshold δ_W. However, since our scheme assumes a very low value for ε_1 (i.e. $\varepsilon_1 = 10^{-9}$), N does not affect δ_W much in practice.

3.4.4 Database Update Attacks

We have summarized the possible database attacks, and discussed how our watermarking scheme can preserve join result in the previous sections, now we brief how these attacks can be defeat-able using our watermarking technique.

A1. Bit Modification

The work in [2] suffers exactly from this attack. Our individualized watermarking technique is resistant to this attack because we apply spread-spectrum watermarking instead of relying on LSB method.

A2. Tuple Reshuffle

It is obvious that with the re-sorting algorithm, the tuples of the relational database can be re-sorted and thus the synchronization can be restored. Therefore, Attack **A2** can be defeated.

A3. Subset Selection

As we maintain the hash table for detection purpose, both the deleted and newly inserted tuples can be recognized. Since detection phase is correlation based, and we embed as many copies of the same watermark as possible in each watermarked copy and utilize a majority-voting scheme, the delete or insert updates do not affect the watermark detection result that much. Therefore, Attack **A3** can also be overcome.

A4. Subset Addition

Similar to Attack **A3**, Attack **A4** can be defeated.

A5. Attribute Projection

Because we can always recognize the attributes we have selected for watermarking, we can easily know if some of the watermarked columns have been dropped. As long as not all the watermarked columns are dropped, we can still detect the existence of individualized watermark. We assume that no attacker will drop all the numeric columns as it will render the relational data significantly less useful. Therefore, Attack **A5** can be defeated as well.

A6. Attribute Permutation

Attack **A6** is similar to Attack **A2**. For the detection purpose, we need to restore the sequence of the attributes in the relation. Since our technique preserves the mean and variance of the data, we can use this information to recognize the watermarked columns. Also, as our watermarking is column-independent, permuting attributes does not really affect the watermark detection.

A7. Collusion Attack

The two forms of collusion attack have been discussed in detail earlier.

3.5 Experimental Study

In this section, we present the experimental results to complement the theoretical analysis in the previous section. Our experiments were performed on the relational databases generated based on the TPC-H Benchmark, which has been designed to evaluate the performance of decision support systems by executing sets of queries against a standard database under controlled conditions. The database generated consists of 8 tables, each of which has a primary key or composite primary keys. The primary key or composite primary keys can be used for the tuple re-sorting purpose. The schema of the tables is illustrated in Appendix. We embed watermark all the tables except for table NATION and REGION, which have no numeric attributes.

We apply the individualized watermark embedding algorithm to these tables. For each buyer, we apply the algorithm with a distinct individualized watermarking sequence. Then we use the detection algorithm to establish the ownership and track malign buyers when piracy occurs.

All our experiments were performed on a PC with Pentium IV 1.6Ghz CPU, 256 MB of memory, and a 20 GB hard disk drive. The watermark embedding and detection algorithm were coded in Java. The feasibility of the watermarked databases against standard databases is verified by Relational DBMS MySQL. As we embed watermark in data domain, the watermark embedding and detection phases can both be run in linear time $O(N)$, where N is the size of host data (i.e. the number of tuples watermarked or the number of tuples involved in the detection phase). In our experiment, we set the length of individualized watermark $L = 100$ and vary the tolerance parameter $\tau = 0.05 \sim 1.0$ for different tables.

3.5.1 Query Validation

First of all, we should be able to show the imperceptibility of our individualized watermarking scheme, that is the watermarked databases should remain useful upon querying. Thus, we validate the watermarked databases by comparing the query results with the query results of standard unmarked databases.

There are twenty-two decision support queries in total as defined in the TPC-H Benchmark. All these queries were executed using the suggested parameters and produced satisfactory output data. If the query does not involve the watermarked column, obviously the watermarked database produces the same output data as the standard database. Even when we query on the watermarked columns, experiments show that output data are of small numerical difference (within the distortion tolerance) from the standard database.

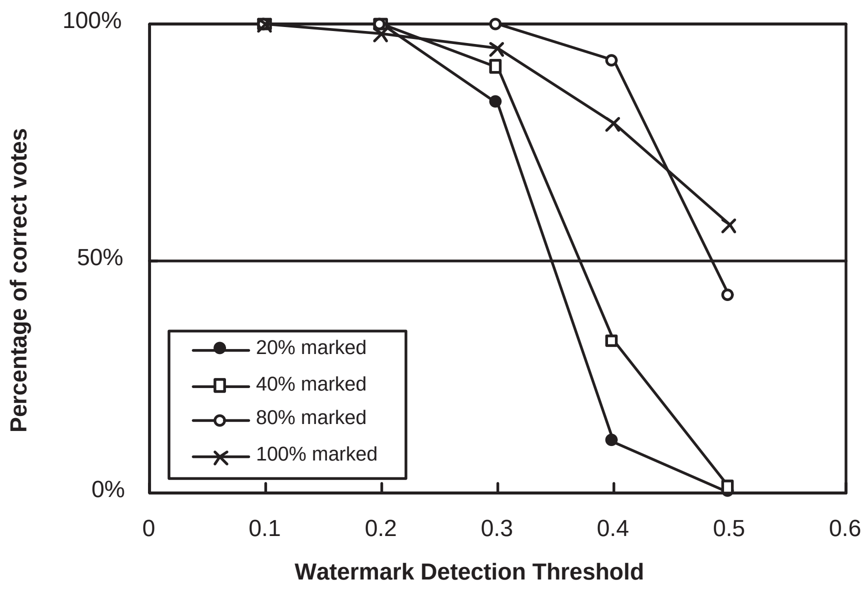

Figure 3.3: Varying the percentage of tuples watermarked

3.5.2 Setting Watermark Detection Threshold

Having established the utility of the watermarked database, we now present the experimental results of robustness of our individualized watermarking algorithm. We present the experimental results of one table out of the 6 tables, i.e. Table SUPPLIER. Table SUPPLIER consists of 10,000 tuples, each with 7 attributes. We embed individualized watermark in the numeric attribute SUPPLIER.ACCTBAL. The parameters used were $L = 100$, $\tau = 1.0$.

Figure 3.3 illustrates the percentage of correct votes with various detection threshold δ_W, under the condition that there are no attacks.

We can see that setting the watermark detection threshold at $\delta_W = 0.3$, we can safely get over 50% correct votes (so as to correctly indicate the presence of the watermark), with percentage of tuples watermarked varying from 20% to 100%.

Figure 3.4 illustrates under the condition that when there are no attacks, the percentage of correct votes with various threshold δ_W, plotted against various chip rate $cr = 20 \sim 100$. From figure 3.4, we can see that setting the threshold $\delta_W = 0.3$, we can achieve more than 50% correct votes.

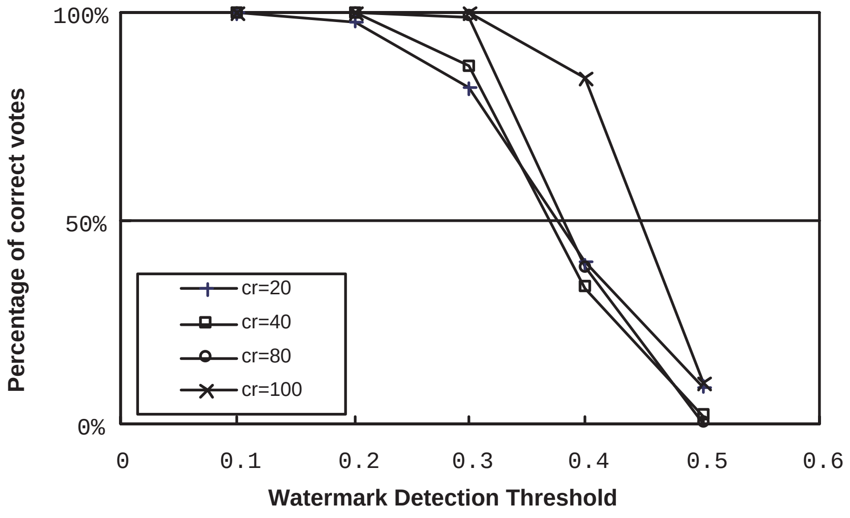

Figure 3.4: Varying the chip rate cr

Figure 3.4 also shows that the percentage of correct votes increases with the increase of the chip rate cr. Therefore, a larger cr is preferred. Given 10,000 tuples and watermark sequence of length $L = 100$, the chip rate can be as large as $cr = 10000/100 = 100$, with the percentage of tuples watermarked reaching 100%. In our following experiments, we set $cr = 100$.

Note that the most damaging failure in the watermark embedding and detection phases is the probability of false positives ε_1, i.e. incriminating a buyer who did not participate in the collusion. From Figure 3.3 and Figure 3.4, we conclude that by setting $\delta_W = 0.3$ (or higher), we can almost drive $\varepsilon_1 = 0$. That is, the proposed method is robust against false positive errors.

In the following sections, we present watermark detection results after various attacks have been applied to the watermarked data. The parameters used are $N =$

10000, $L = 100$, $cr = 100$ and $\tau = 1.0$.

3.5.3 Performance Against Attacks

Subset Selection Attack

Figure 3.5 presents the percentage of correct votes obtained when various percentages of tuples have been randomly deleted. Note that our scheme defines a secret order for the tuples, such that we can recognize which tuples have been deleted and which are newly-inserted. In both cases, these tuples are not involved in the calculation of correlation values. Therefore, the percentage of correct votes is affected little by the increase in the percentage of deleted tuples. We can see that even with up to 90% of tuples deleted, we can still detect the watermark with the watermark detection threshold of $\delta_W = 0.3$.

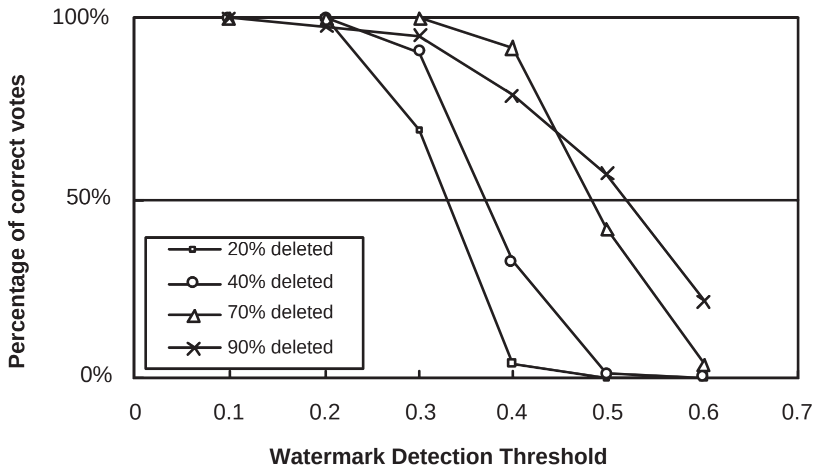

Figure 3.5: Varying the percentage of tuples deleted

Similarly, with the insertion of new tuples or attributes, the correlation value will not be affected either, since we always know which tuples or attributes are new and thus do not involve them in the correlation calculation. Therefore, even if the attackers insert a large number of tuples or attributes, our detection scheme will still work. However, it is obvious that too many insertions of useless tuples or attributes will make the relational data lose value, which is not desirable from the attackers' point of view.

Project Operation Attack

Often, the attackers project only a subset of watermarked columns. As there are too small number of columns available for watermarking in Table SUPPLIER and the other tables generated by TPC-H Benchmark, we experiment Project Operation Attack on another dataset named Ovarian Cancer [1]. The dataset consists of 253 instances (tuples), each with 15,154 attributes (all numeric). We add an extra attribute (column with all the values distinct from each other) to serve as the primary key for the purposes of re-sorting tuples. We select 100 numeric attributes as candidates to be watermarked.

We can consider the Project Operation Attack equivalent to deleting columns. The fewer columns we project, the more columns are considered to be deleted. As long as not all of the 100 watermarked attributes are deleted, the individualized watermark can still be detected from the remaining columns.

Figure 3.6 presents the percentage of correct votes with various percentages of watermarked columns deleted. Again, we can see that the watermark detection phase works well with up to 90% of the attributes deleted.

Collusion Attacks

As we proved in the previous section, with a small collusion clique size, the attackers will fail to eliminate the watermark. On the other hand, it is almost impractical to

[1]National Cancer Institute, http://clinicalproteomics.steem.com/

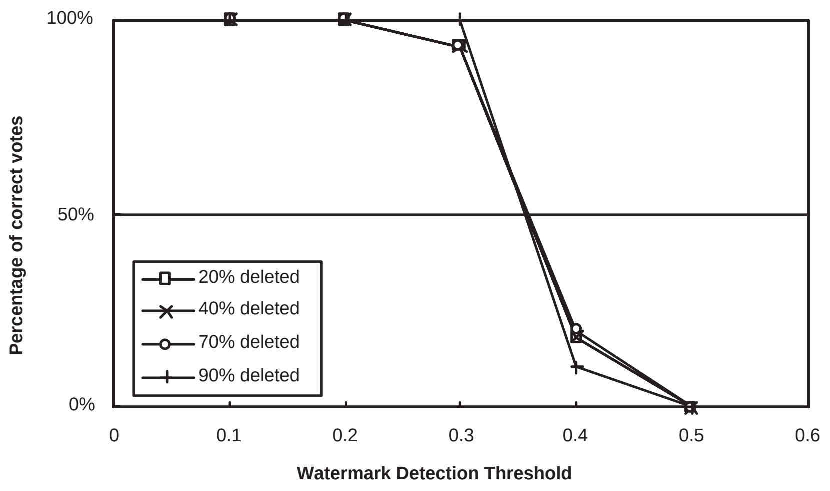

Figure 3.6: Varying the percentage of watermarked columns deleted

achieve a large collusion clique.

With a collusion size of $K = 5$, we can see that the attacker's copy obtained from averaging collusion attacks still shows significant correlation with the detection keys in the collusion clique, namely $\{w_i, i = 1, 2, ..., 5\}$.

Figure 3.7 shows the correlation median of the attacked data over the 5 colluded detections keys. We compute the median of correlation values because if the median is greater than the detection threshold δ_W, then the percentage of correct votes is over 50%, which is good enough to confirm the presence of the individualized watermark.

We can observe that even after averaging $K = 5$ different copies of the relational database, the correlation median values show no apparent decrease and mostly remain greater than the detection threshold, except for the correlation median of *colluder 3* with 20% tuples watermarked. But as long as we can find the majority of the colluders within a collusion clique, it is highly likely that we can get to know the whole clique. Therefore, with our individualized watermark embedding and detection technique, the colluders can be precisely and reliably identified.

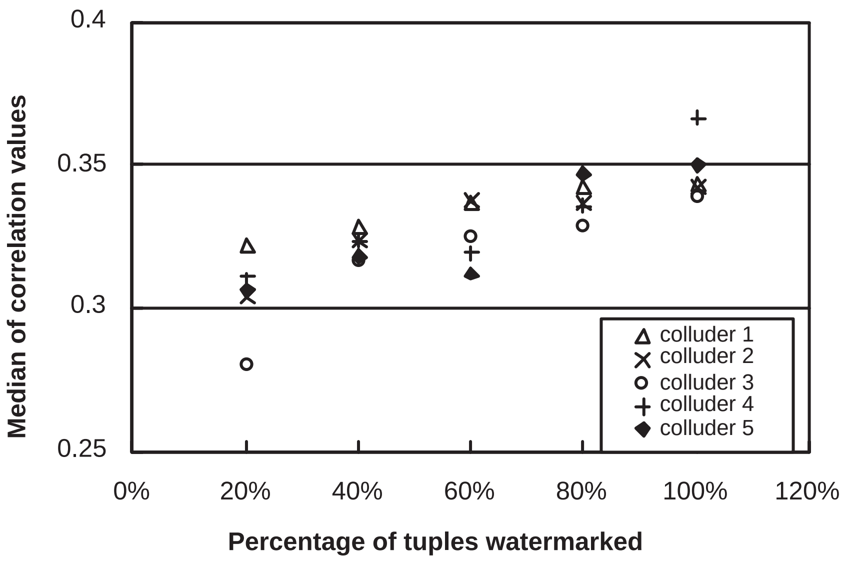

Figure 3.7: Median of correlation values after collusion attack of size $K = 5$

Even after increasing the collusion clique size to $K = 10$, we could still detect the watermark. This demonstrates that our technique is indeed robust against the collusion attack.

3.6 Summary

In this chapter, we have addressed the problem of protecting digital rights of relational data under the buyer-seller scenario where the same data are sold to several buyers. We have articulated the need for collusion resistance and thus the need for individually fingerprinting each copy of the data sold. We then proposed a novel individualized watermarking method in which we utilized a direct sequence spread-spectrum (DSSS) based approach for distinctly watermarking every copy of the data. The proposed mechanism can survive a wide variety of attacks, such as data re-sorting, subset selection, subset addition etc., as well as collusion attacks. We gave a formal security analysis as well as the experimental results on a real-life dataset to show the robustness of our technique against various attacks. The validation of the join results shows that the distortions introduced by our technique are within the tolerance of the relational data.

Our scheme supports up to 2^L distinct buyers. When the number of buyers is small, a general pseudo-random number generator works well. However, when the number of buyers increases, it is necessary to use good pseudo-random sequences possessing certain properties, in terms of sequence length, auto-correlation, cross-correlation, orthogonality, and bit balancing. For these concerns, good pseudo-random sequences, such as M-sequence and Gold sequence [28] can be used.

Chapter 4

Invertible Watermarking of Databases

In many applications, i.e. medical, military, satellite etc., it's important for a legitimate user to verify the integrity of the data before using it, because inaccuracy in these data will not only render the data useless but result in vicious effects. Inaccurate data are unacceptable, while free redistribution of the data should be prohibited. Unfortunately, most approaches of copyright protection modify the data to some extent. Therefore, when precision-critical data are sold, a trade-off must be achieved between precision and integrity of the data on one hand and copyright protection on the other. In this case, traditional watermarking as a currently wide-used solution for protecting copyright is incompetent, and invertible (or lossless, reversible) watermarking is therefore required. Invertible watermarking has been introduced in the literature for image watermarking and authentication. In this chapter, we present an invertible watermarking scheme for precision-critical databases. The data can be recovered fully or partially with the legitimate users presenting the secret key, while being totally meaningless to the non-privileged users.

This chapter is organized as follows. Section 4.1 gives the problem specification of this chapter. Section 4.2 introduces a naive LSB approach to invertible watermarking of databases. Section 4.3 presents a more decent spread-spectrum based invertible watermarking of databases. Section 4.4 addresses multi-level access. Experimental study is given in Section 4.5. Section 4.6 summarizes the chapter.

4.1 Problem Specification

Watermarking as a most widely used solution for copyright protection has been developed well for image, audio and video etc. signals. Relatively few works have been published for database watermarking. [2, 30] present idea of watermarking relational database respectively. In the previous chapter, we gave out a novel watermarking scheme that can trace pirate and survives collusion attacks. All these work [2, 30] are based on watermarking techniques that modify the data values more or less, and recovery of the original data is impossible. For example, in [1], the authors assumed:

> "the decrease in the value of the data is small enough that the owner is willing to pay this price in exchange for that ability to assert ownership."

However, for some applications, such as medical, military and satellite information, the integrity of the data is important. Reasons are obvious: medical data are used for disease diagnosis, and any inaccuracy can potentially harm the patient and can even lead to death. Similarly, satellite data are used to locate military strategic targets or to forecast weather, thus inaccurate data may be totally useless. In [14], an invertible watermarking for JPEG images authentication is presented. The assumption that original images have been lossy-compressed is a drawback. Even after the watermark is removed, the result is still lossy-compressed image. In [13], Fridrich et al improved the idea and applied it to watermark all image formats. [9] present a spread-spectrum invertible watermarking system for authenticating images in lossless formate. The integrity of the images can be verified and original images before embedding watermark can be recovered. [10] also proposed a spread-spectrum watermarking scheme for rights management of broadcast video. The methodology in [10] and [9] are similar to some extent. Philosophically, we would like to adopt this idea of invertible watermarking to watermark precision-critical database.

However, invertible watermarking of databases has been made more difficult than invertible watermarking images for the reasons that have been discussed in chapter 3. The particular properties of databases must be taken into account when watermarking the relations.

For a probable application of invertible watermarking databases, we assume that different users may have different levels of access. Multi-level access can be defined in such a way that:

1. General users: can only see watermarked version of the data.
2. Read-only users: are allowed to remove the watermarks partially or fully (defined by the different levels of clearance).
3. Administrators: not only can read the watermark-free data, but also can update the data.

Assumption: Read-only users are a small group of users to whom the owner of the data trusts to some extent. Administrators have permission to update the data, and delete and insert items, and are even fewer in number.

In this Chapter, we take two different approaches for invertible watermarking databases: One is a LSB watermarking technique, and the other is a spread-spectrum technique.

4.2 A Naive Approach

In this section, we present an invertible watermarking technique based on LSB (Least Significant Bit).

Given a database tuple t with schema $(P, A_1, A_2, ..., A_\nu)$, let $t.P$ denote its primary key. Without loss of generality, assume that $A_1, A_2, ..., A_k$ are candidates for watermarking, where $A_i, 1 \leq i \leq k$ are numeric attributes.

Let $LSB_n(A_i)$ be the n^{th} least significant bit of A_i. Here we assume that we only modify $LSB_1(A_i)$. We use a standard cryptographic hash function to determine the watermark bits for tuple t. Let the hash value be

$$F(t.P) = H(K \circ H(K \circ t.P)),$$

where K is a secret key, and H is a hash function like SHA-1 or MD-5.

The watermarking rules are as follow:

H1: $t.A_i$ remains intact, if $F(t.p)$ is even;

H2: $LSB_1(t.A_i)$ is flipped, otherwise.

Because we keep the hash function H only known to the privileged users, non-privileged users do not know which LSBs have been flipped.

In the watermark removal phase, we compute hash value again, and recover the original data as follow:

H1: $t.A_i$ remains intact, if $F(t.p)$ is even;

H2: $LSB_1(t.A_i)$ is flipped, otherwise.

We can observe that the watermark removal is just the same as the watermarking phase. This naive approach based on LSB technique is simple and effective with high accuracy. However, it may not be a perfect solution to protect the precision-critical databases, because its security entirely relies on the security of the hash function, and once the hash function is disclosed or tried out, the data will be fully exposed to piracy. Therefore, in the next section, we will present a scheme that is more elegant and secure.

4.3 Spread-spectrum Approach

Given the deficiency of the naive approach, in this section we discuss how spread-spectrum watermarking technique can be used for invertible watermarking databases.

4.3.1 Preliminary

First, we propose the fundamentals of the spread-spectrum watermarking technique, and how it can be invertible.

Watermark Embedding

The original data signal is x_i, $i = 0, 1, 2, ..., N$, and the copyright information is a binary sequence $a_j \in \{-1, 1\}$, where $j = 0, 1, 2, ..., M$. We spread a_j with chip-rate cr to obtain the spread sequence b_i as follows:

$$\forall j\colon\ b_i = a_j,\ j \cdot cr \leq i < (j+1) \cdot cr,$$

where cr is a large factor selected in such a way that $cr \times M = N$. The spreading provides redundancy of the watermark information and improves the robustness in watermark extracting and removal phase. Then the spread sequence is multiplied with a pseudo-random noise sequence $\{p_i\}$ and the amplitude (also called amplitude or tolerance factor α_i, where $p_i \in \{-1, 1\}$, and $\alpha > 0$, such that the watermarked signal is computed as

$$y_i = x_i + \alpha_i \cdot p_i \cdot b_i.$$

Watermark Detection

Watermark inversion (detection and removal) is performed by demodulating the watermarked signal y_i with the same pseudo-noise sequence p_i used in the embedding phase. The original unwatermarked data is not required for the detection. We multiply the watermarked signal y_i by the pseudo-noise sequence p_i over each window of the embedded watermark information. Let $\widehat{y_i}$ be probably modified version of y_i. In fact, in the applications proposed in this chapter, y_i is not likely to be maliciously changed to $\widehat{y_i}$ by non-privileged users as they do not have access right and any improper update may render the data useless. Correlation summation s_j is calculated over each window of the embedded watermark information a_j as following:

$$\begin{aligned} s_j &= \textstyle\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i \cdot \widehat{y_i} \\ &= \textstyle\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i \cdot x_i + \sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i^2 \cdot \alpha_i \cdot b_i \\ &\approx \textstyle\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i^2 \cdot \alpha_i \cdot b_i. \end{aligned}$$

Note that $\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i \cdot x_i = 0$ if pseudo-noise sequence p_i and original signal x_i are uncorrelated.

The extracted watermark information $\widehat{a_j}$ (where extracted $\widehat{a_j}$ may be different from the original a_j due to distortion or inevitable correlation between p_i and x_i for real data), can be interpreted by the sign of s_j, as

$$\begin{aligned} sign(s_j) &= sign(\textstyle\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} p_i^2 \cdot \alpha_i \cdot b_i) \\ &= sign(\textstyle\sum_{i=j\cdot cr}^{(j+1)\cdot cr-1} \alpha_i \cdot a_j) \\ &= sign(a_j \cdot \textstyle\sum \alpha_i / cr). \end{aligned}$$

Since amplitude factor α_i is a positive number, we have

$$sign(s_j) = a_j.$$

Thus, the embedded watermark information a_j can be extracted from the correlation summation with high probability of correctness.

Note that the amplitude factor α_i used in embedding phase must also be known. To easily meet this requirement and simplify the calculation, we set α_i to a constant value α.

Watermark Removal

We can then remove the watermark from the data to achieve the watermark-free data as follows:

$$\widehat{x} = \widehat{y} - \alpha \cdot b_i \cdot p_i.$$

Obviously, if the embedded bits a_j are correctly recovered (when $\widehat{a_j} = a_j$), the watermark-cleaned data will match the original data, i.e. $\widehat{x} = x$, if $\forall i$, $\widehat{a_j} = a_j$.

Note that the pseudo-noise sequence p_i and amplitude factor α (or α_i) must be known in order to remove the watermark and view the watermark-free data. Only the privileged users (Read-only users and Administrators) can know p_i and α_i. Non-privileged users (Guests) have no knowledge of p_i and α_i, and thus have no access to the original data.

4.3.2 Invertible Watermarking of Relational Databases

The above mentioned method can work well for image watermarking, but is not directly applicable to watermarking of databases. The problems to be solved include the lack of ordering of the tuples and the update, delete and insert of the tuples.

Figure 4.1 shows the process of watermarking embedding and removal.

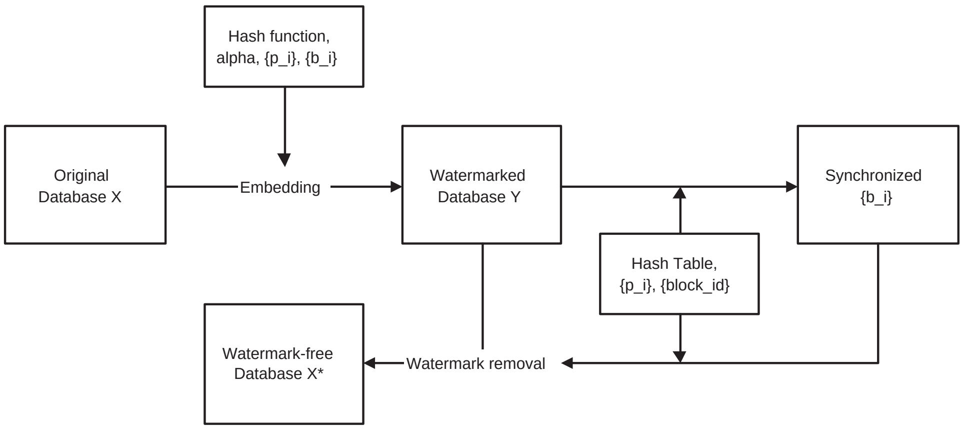
Hash function, alpha, {p_i}, {b_i}
Original Database X
Embedding
Watermarked Database Y
Synchronized {b_i}
Hash Table, {p_i}, {block_id}
Watermark-free Database X*
Watermark removal

Figure 4.1: Watermarking Embedding and Removal

Sorting Database Tuples

In image watermarking, pixels have fixed relative positions. Unfortunately, the tuples of a relational database has no defined order. However, a fixed order is necessary to detect the embedded watermark and keep the detection phase synchronized with the embedding sequence. Thus, some invariant information in databases can help define such an order. For our technique, we choose the primary key information (for example tuple ID) for the re-sorting purpose. Note that in our application assumption, only Administrators have access to remove or update the tuples, and Administrations will not maliciously modify the primary key information.

Let R be a database relation with schema $R(P, A_1, A_2, ..., A_\nu)$, where R is the primary key. We use a standard cryptographic hash function like SHA-1 or MD-5 for the purpose of establishing order [31], because the hash function has desirable properties such as: (1) They are one-way function, i.e. given a hash value, it is computational impossible to find the corresponding input. (2) It is computational impossible to find two inputs that yield the same hash value. With these two properties, the primary key information can be hashed to define a unique order of the tuples of the relational data. The hash value $F(t \cdot P)$ of primary key attribute P of tuple t is computed using:

$$F(t \cdot P) = H(K \circ H(K \circ t \cdot P))$$

where K is a secret key, H is a cryptographic hash function, and $\circ$ denotes concatenation. We maintain the hash table for the watermark detection needs.

Note that our method is not constrained to single table query, as it can also be applied to complex queries across multiple tables. However, the primary key and foreign keys should not be absent from watermarking, since improper modification of these attributes will introduce great errors to join result.

Embedding Phase

In [2, 30], only the numeric attributes can be watermarked. But in our approach, both numeric and non-numeric attributes can be used to embed watermark. Let the original relation R have schema $R(P, A_1, A_2, ..., A_\nu)$. Without loss of generality, assume the first k attribute $A_1, A_2, ..., A_k$ are the candidates to be watermarked. For the sake of simplicity, in the algorithm below we only watermark attribute A_1, i.e. $x_i = A_{1,i}$. To watermark more attributes, we can simply apply the embedding algorithm to these attributes respectively.

Figure 4.2 presents the invertible watermarking algorithm.

We embed the watermark bits b_i, $i = 0, 1, 2, ..., N$, combined with the pseudo-noise

p_i and amplitude factor α into the original data x_i, according to the tuples order defined in the above re-sorting algorithm,

$$y_i = x_i + \alpha \cdot p_i \cdot b_i.$$

Note that the pseudo-random number sequences p_i is generated from some

For simplicity of calculation, we assume that for all the columns to be watermarked in one table, we use the same pseudo-noise sequence p_i and b_i. Note that multiple versions of b_i can be embedded into the original data, given different orthogonal p_i sequences. We replicate copyright information a_j number of cr times, to get a block of $b_i = a_j$ spreading the watermark bits. This nature of spread-spectrum watermarking improves the robustness of the watermarks (against attacks) and leads to high probability of correctness in watermark extraction phase with low distortion of attribute data values.

Invertible Watermarking Embedding:

Input: original data x_i, secret key sk, binary sequence a_j, $j < i$, amplitude factor α
Output: watermarked data y_i
Method:

```
1: Generate pseudo-noise sequence $p_i$ with secret key $sk$;
2: Duplicate $a_j$ with chip-rate $cr$ to generate a binary sequence $b_i = a_j$, $j \cdot cr \leq i < (j+1) \cdot cr$;
3: for each $i \leftarrow$ to $N$ do
4:     Compute the new value $y_i = x_i + \alpha \cdot p_i \cdot b_i$;
5: return $y_i$
```

Figure 4.2: Invertible Watermarking Embedding

Table 4.1 outlines the information we need to keep for watermark extraction and removal phase. The information includes *Hash* of the private key, its corresponding p_i and the watermarking window ID *bid*. Assume the window width (chip-rate) is $cr = 50$.

Hash value	p_i	b_{id}
10000000000000000000	1	1
10000000000000000001	-1	1
10000000000000000010	-1	1
...	...	...
1000000000000110011	-1	2
1000000000000110100	1	2
1000000000000110101	1	2
...	...	...
1000000000001100101	1	3
1000000000001100110	-1	3
1000000000001100111	-1	3
...	...	...

Table 4.1: Maintained Hash Table

It is secure to keep p_i and *bid* along with $Hash$ of the private key, for the reason that $Hash$ is an one-way function. Given the output $Hash$, it is computational infeasible to compute the input, that is the private key P. Therefore, even if the Hash Table is disclosed to non-privileged users, they will not be able to find out how p_i and *bid* are matched to the private key P, thus can not remove the watermark and view the original data.

Extraction Phase

The database tuples can be deleted and inserted frequently, and the tuple ordering loses synchronization with pseudo-noise sequence p_i and binary sequence b_i. To be able to extract the watermark and recover the original data without loss, the synchronization is a must.

Figure 4.3 outlines the invertible watermark extraction algorithm. The algorithm consists of two phases. Phase 1 re-sorts the tuples of watermarked data to synchronize with p_i and b_i. Phase 2 scans all the tuples in re-sorted order, and recover the watermark bit $\widehat{a_j}$, where $\widehat{a_j}$ is the recovered version of a_j. Obviously, we aim at $\widehat{a_j} = a_j$. The following paragraphs describe the three phases in more detail.

Invertible Watermarking Removal:

Input: secret key K and sk, watermarked data $\widehat{y_i}$, chip rate cr
Output: extracted watermark bit $\widehat{a_j}$, watermark-free data $\widehat{x_i}$
Method:

phase 1: database re-sorting

```
1: for each tuple $t$ in the database do
2:     Compute $F(t \cdot P) = H(K \circ H(K \circ t \cdot P))$;
3:     Scan the kept Hash table and restore its order by $F(t \cdot P)$;
```

phase 2: watermark removal and data recovery

```
1: for each tuple $t$ in the re-sorted order do
2:     for each watermark block $j \leftarrow 0$ to $M$ do
3:         for each $i \leftarrow j \cdot cr$ to $(j+1) \cdot cr - 1$ and $bid = j$ do
4:             Compute $a_j = c_j = \sum_{i=j \cdot cr}^{(j+1) \cdot cr - 1} p_i \cdot \widehat{y_i} \approx \sum_{i=j \cdot cr}^{(j+1) \cdot cr - 1} p_i^2 \cdot \alpha_i \cdot b_i$;
5:             Remove the watermark $\widehat{x_i} = \widehat{y_i} - \alpha \cdot p_i \cdot a_j$;
```

Figure 4.3: Invertible Watermarking Removal

Note that only the watermarked data are stored in the DBMS. We assume the recovered relation is treated as a snapshot of the relational databases. Note that the view could be created only based upon a subset of the whole data. And the view will be destroyed compulsorily after the legitimate user logs off.

Incremental Watermarking

One common concern of watermarking databases is the cost in terms of time and space. In many applications, databases are updated frequently. Therefore, watermarking is required to be done frequently as well. Intuitively, the overhead can be greatly reduced by *Incremental Watermarking*: If only a small portion of the databases (eg. a number of tuples) are updated, we re-watermark that small portion only. Incremental watermarking does not make much sense for watermarking image or other multimedia contents, as the images or multimedia signals are not to be updated very often. However, it is of great importance for watermarking databases as databases are updated frequently and the cost must be carefully considered and reduced to utmost.

We assume that all the updates to database are applied on watermark-free (watermark removed) data (the view) instead of on the watermarked data. It makes sense because only *Administrators* are granted the access to update the data, and *Administrators* have access to read the watermark-free data. To update on a watermarked version of the data will only make the process complex and the data in a mess and irreversible. Incremental watermarking is done afterward based on both the watermarked version and the updated original data.

Figure 4.4 outlines **five** cases of database updates, namely *update*, *insert tuple* and *delete tuple*, *insert column*,*delete column*. These five cases are dealt with differently as *Case 2* and *Case 3* change the ordering of the tuples if re-sorted, while the others do not.

For *update*, only the attributes values are changed, so we can merely watermark this tuple again with the corresponding p_i and b_i, and modify the watermarked values correspondingly in the watermarked databases.

4.4 Multi-level Access

As mentioned in the introduction, different classes of users (Guests, Read-only users, and Administrators) can be granted different levels of access (multi-level access). Users are classified into three classes:

1. General Users: can only see watermarked version of the data.

```
Incremental Watermarking Algorithm:
Input: updated data $\widehat{x_i}$, amplitude factor $\alpha$, recovered $\widehat{b_i}$ if Case 1 and Case 4
Output: e-watermarked data $\widehat{y_i}$
Method:

Case 1: Update data value
  1: Hash the tuple and retrieve corresponding $p_i$;
  2: Re-watermark this tuple $\widehat{y_i} = \widehat{x_i} + \alpha \cdot p_i \cdot b_i$.

Case 2: Insert new tuple
  1: Hash the new tuple and add it the end of hash table, generate additional $p_i$ and $b_i$;
  2: Watermark this tuple $\widehat{y_i} = \widehat{x_i} + \alpha \cdot p_i \cdot b_i$.

Case 3: Delete one tuple
  1: Delete its Hash, $p_i$, and $bid$ from the Hash table;
  2: Delete its corresponding tuple from the watermarked table.

Case 4: Insert new column
  1: Hash the column and retrieve corresponding $p_i$;
  2: Watermark the whole column $\widehat{y_i} = \widehat{x_i} + \alpha \cdot p_i \cdot b_i$.

Case 5: Delete one column
  1: Delete corresponding sequence $p_i$, if $p_i$ is not applicable to other columns;
  2: Delete its corresponding column in watermarked table.
```

Figure 4.4: Incremental Watermarking Algorithm

2. Read-only users: are allowed to remove the watermarks partially or fully (defined by the different levels of clearance). With the highest level of clearance, the user can view the whole watermark-free data.

3. Administrators: can not only read the watermark-free data but also update the data.

For example, in medical applications, Administrators can be the doctor who can read and write (even modify) the data and prescription, Read-only users can be the nurses who read the data and follow instructions, and General users can be other people outside the above two classes. In military application, Administrators can be commanders who make strategies, Read-only users can be soldiers entitled to know the strategies, while General users can be those from a third party (even the enemies).

Note that Read-only users are granted access to the data to the extent of their clearance. At one extreme end, users with no clearance at all (i.e. Guests) can only view the watermarked data, which appears good but is not suitable for high-precision processing. At the other extreme end, users with full clearance (for example, high-level Read-only users and Administrators) can invert and remove all the watermarks so as to obtain the original data. In between are users with intermediate clearance, who can only remove parts of the watermarks. A simple solution to this problem is to watermark fractions of the database with different pseudo-random sequences (treated as secret keys) which are respectively known only to specific groups of users. The users with high-clearance get to know a set of pseudo-random sequences. While this is not the main contribution of the work, we do not discuss it in further detail.

4.5 Experimental Study

In this section, we present the experimental study and analyze the cost. Our experiments were performed on the dataset named *Ovarian Cancer* [1]. The dataset consists of tables with various instances (tuples) and attributes. We choose one table bearing 15,154 instances (tuples) and 2 columns (attributes). We add an extra attribute (a column with all the values distinct from each other) to serve as the primary key for the purposes of re-sorting tuples.

We ran our experiments on Windows XP with a P4 1.6GHz Intel processor, 256 MB of memory, and a 20 GB hard disk drive. The watermark embedding, extraction and

[1]National Cancer Institute, http://clinicalproteomics.steem.com/

removal algorithms were coded in C Language. As we embed watermark in spatial domain, the watermarking and detection scheme can both be run in linear time $O(n)$, where n is the number of tuples involved in watermarking and detection scheme. In our experiment, we set the parameters empirically.

4.5.1 Setting Parameters cr and α

Given the invertible watermarking scheme discussed above, we expected the extracted $\widehat{a_j}$ equal to embedded watermark information a_j, thus to achieve $\widehat{x_i} = x_i$. Here, we experiment on the precision-critical data to see how a_j vs $\widehat{a_j}$ and x_i vs. $\widehat{x_i}$ match.

Figure 4.5 shows setting $\alpha = 0.5$, how the chip rate cr affects the percentage of successful watermark bits detection.

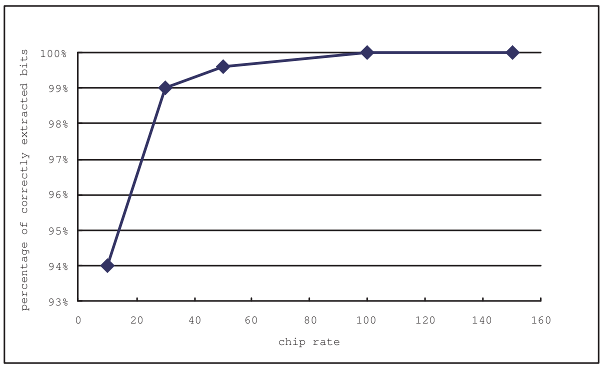

Figure 4.5: Varying chip rate

We can see from Figure 4.5 that given $\alpha = 0.5$, the percentage of correctly detected watermark bits increases as the chip rate cr gets larger. Note that for our application, it is compulsory to recover every watermark bit exactly as it was, so that the percentage of correctly detected watermark bits must be 100%. We repeated the detection phase 50 times, each time with a set of different seeds for the pseudo random sequence $\{p_i, i = 0, 1, ..., N-1\}$ and $\{a_j, j = 0, 1, ..., M-1\}$, and experiment results show that setting $cr = 100$, we can safely ensure 100% watermark bits to be detected correctly.

Next, fixing $cr = 100$, we choose the value of α. Figure 4.6 shows how the percentage of correctly detected watermark bits varies with *alpha*.

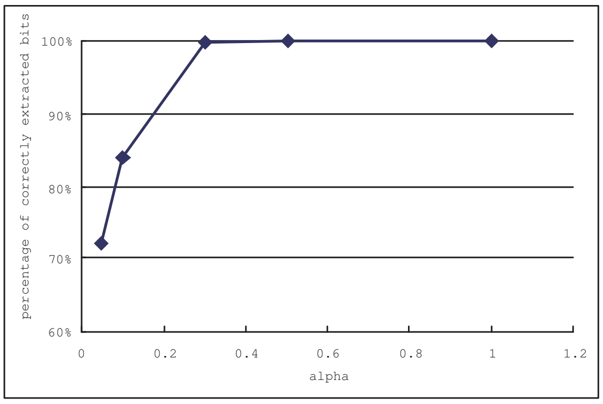

Figure 4.6: Varying alpha

We can see that the percentage of correctly detected watermark bits increases with the increase of α. Set $\alpha = 0.5$ 1.0, we can achieve percentage of correctly detected watermark bits of 100%.

In the following test, we set $cr = 50$ and $\alpha = 0.5$.

4.5.2 Varying Percentage of deleted tuples

Without update, delete and insert operations, the precision-critical database can be correctly inverted. However, the update, delete and insert operations can not be neglected, as they are likely to mess up the ordering of the re-sorted tuples, make tuples lose synchronization with the watermark sequence, and introduce error to the watermark removal phase.

We can see from the earlier discussion that as long as the privileged user has access to the maintained hash table, insert of one tuple will not make the watermark removal phase too difficult. However, delete and update will make the watermark not integrity.

On one hand, since the watermark detection phase is spread-spectrum based, it can survive tuple (watermark) deletion to some extent.

Figure 4.7 plots how the percentage of wrongly detected watermark bits goes up with the increase of number of deleted tuples.

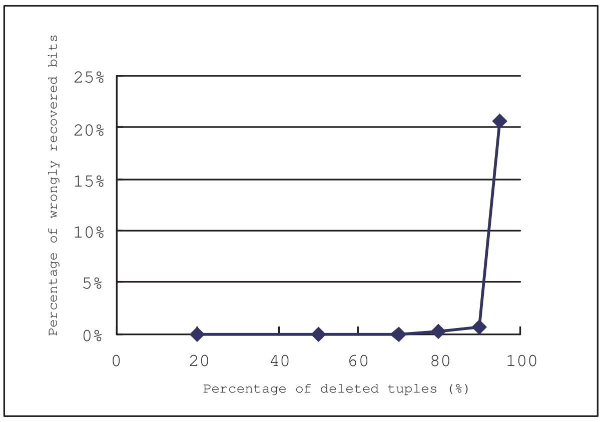

Figure 4.7: Effect of Incremental Watermarking on Running Time

We can see that when the percentage of randomly deleted tuples reaches 70% and above, the watermark bits can no longer be completely correctly detected. It means the spread-spectrum based watermark detection can survives up to 70% tuples randomly deleted while preserving the integrity of the watermark.

On the other hand, because normal database update may change the values greatly, the embedded watermark may be severely destroyed. With a small portion of the tuple values updated, depending to what extent the values have been varied, the watermark detection can be totally wrong. Therefore, we suggest re-watermarking the databases after updates have been issued. While re-watermark can be very expensive given the frequency of update operations, our *Incremental Watermarking* scheme can cleverly solve this problem. Next, we presents the performance of our invertible watermarking scheme against varying update percentage.

4.5.3 Effect of Incremental Watermarking

Here, we compare the computation cost of re-watermarking the whole database with the cost of incremental watermarking. We compare the running time of these two watermarking methods, varying the percentage of UPDATES introduced to the databases. Without loss of generality, UPDATES includes updating values, inserting tuples and deleting tuples only, because the proof of efficiency of incremental watermarking over inserting and deleting columns is trivial.

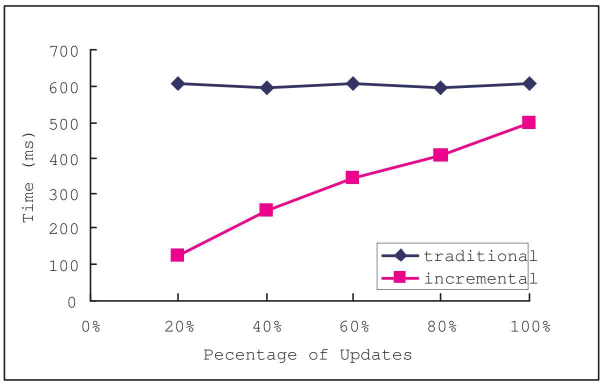

Figure 4.8: Effect of Incremental Watermarking on Running Time

Figure 4.8 shows that when the percentage of UPDATES is below 80% of the whole databases, incremental watermarking always outperforms conventionally re-watermarking the whole databases. When the percentage of UPDATES reaches 80% and above, conventional watermarking performs better. This can be easily understandable because searching in the maintained hash table can be time-consuming, and when too many updates occur, it is slower to search and watermark a part than to simply re-watermark the whole.

Therefore, incremental watermarking is very much recommended when a small portion (not necessary to be 80% for different databases, and it can be determined by the application and the database administrator) of the databases is changed. Incremental watermarking has made our invertible watermarking technique more suitable for real-life applications.

4.6 Summary

In this chapter, we have addressed the problem of invertible watermarking precision-critical databases. Given the need for lossless recovering the watermarked data for military or medical uses, we have presented an invertible spread-spectrum watermarking method in which we utilize a direct sequence spread-spectrum based approach for watermarking the data. With our watermark extraction method, the watermark can be removed, and thus the original data can be recovered. We tuned the parameters (α and cr)in the experiments, and achieved the percentage of correctly recovered data of 100%.

Also, with *Incremental Watermarking*, it is no longer necessary to re-watermark the whole databases, when only a small portion of the data are updated. With incremental watermarking, the computation cost of updates and re-watermarking can be greatly reduced. Experimental results have shown the effectiveness of incremental watermarking.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we proposed two novel techniques for watermarking relational databases, each with a different purpose.

Firstly, we present *Individualized Watermarking* of relational databases, where both ownership establishment and traitor tracing were enabled, in that merely proving the ownership without identifying the traitor can not compensate the data owner's loss. We proposed a spread-spectrum based watermarking scheme, which was resilient to various attacks, such as data re-sorting, subset selection, subset addition etc., as well as collusion attacks. The individualized watermarks are collusion-resistant, such that even the attempt of a few buyers colluding together to destroy the individualized watermarks will not be successful. Formal security analysis was given, and the experiment results conducted on TPC-H benchmarked databases were given in support of the analysis. Both the analysis and the experiment results show that the individualized watermark can withstand a variety of malicious attacks and benign updates, thus verify the feasibility and reliability of the individualized watermarking scheme.

Secondly, we proposed *Invertible Watermarking* of precision-critical databases, taking into account that in many database applications, i.e. medical, military, satellite and so on, the accuracy of the data is as important as the copyright protection of the data, because inaccuracy in these data will not only render the data useless but result in vicious effects, while free redistribution of the data will cause great capital loss to the data owner. In this thesis, we discussed two different approaches of invertible watermarking databases, i.e. LSB and spread-spectrum. We abandoned LSB approach because of its deficiency and chose spread-spectrum for its higher security and robustness. Also, we introduced the idea of hierarchical access of the data, and enabled the function of *Incremental Watermarking* when only a small portion of the databases were modified or updated. The experimental study showed that the precision-critical data can be fully recovered without error for the privileged users, while being of no value for non-privileged ones. *Incremental Watermarking* outperformed conventional re-watermarking with percentage of updates up to 80%. The effectiveness of incremental watermarking had made the technique more practical in real applications.

5.2 Contributions

Naturally, the major contributions of this work fall into two aspects:

First, it identified the importance of right management of relational database through traitor-tracing, and enunciated possible attacks that individualized watermark inserted in relational databases must survive. It also proposed a novel individualized watermarking technique geared for relational databases, supported by formal security analysis as well as extensive experimental results.

Second, it articulated of the need for invertible watermarking precision-critical data, and first proposed an invertible watermarking technique with multi-level access. With privileged access to the database, watermark removal was very reliable. Also, it enabled incremental watermarking, which greatly improved the efficiency of re-watermarking a modified databases.

5.3 Future Work

In future work, we would like to address the issue of protecting the buyer's digital rights. Though in most watermarking scenarios, it is assumed that the seller's rights are more important which need to be protected against violation by malicious buyers, yet it may also happen that the seller may be malicious and may try to falsely implicate an honest buyer. Therefore, it is important to come up with reliable solutions to protect both the seller and buyers.

Furthermore, we would like to extend our work to watermark non-numeric attributes, and improve the efficiency of the watermarking schemes. Efficient watermarking is important as to be suitable for watermarking high dimensional databases. In fact, many real applications are devised for high dimensional data, and thus the efficiency is of most concern.

Bibliography

[1] R. Agrawal, Peter J. Haas, and J. Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal*, 2003.

[2] R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proc. 28th International Conference on Very Large Data Bases (VLDB 2002)*, 2002.

[3] M. J. Atallah, V. Raskin, M. Crogan, C. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Proc. 4th International Information Hiding Workshop, LNCS 2137*, 2001.

[4] E. Bertino, M. Braun, S. Castano, E. Ferrari, and M. Mesiti. Author-x: A java-based system for xml data protection. In *Proc. IFIP Workshop on Database Security*, pages 15–26, 2002.

[5] E. Bertino, S. Jajodia, and P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2):101–140, 1999.

[6] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. 1998.

[7] I. Cox, M. Miller, and J. Bloom. Digital watermarking. *Morgan Kaufman Publishers*, 2002.

[8] R. Dixon. *Spread Spectrum Systems with Commercial Applications*. John Wiley, 1994.

[9] Josep Domingo-Ferrer and Francesc Sebe. Invertible spread-spectrum watermarking for image authentication and multilevel access to precesion-critical watermarked images. In *Proc. Of the International Conference on Information Technology: Coding and Computing (ITCC'02)*, 2002.

[10] S. Emmanuel and M.S. Kankanhalli. A digital rights management scheme for broadcast video. *ACM Multimedia Systems Journal*, 2003.

[11] F.Ergun, J. Kilian, and R. Kumar. A note on the limits of collusion-resistant watermarks. In *Eurocrypt*, 1999.

[12] A. Fiat and T. Tassa. Dynamic traitor tracing. In *Crypto, LNCS, vol.1666*, pages 354–371, New York: Springer–Verlag, 1999.

[13] Jessica Fridrich, Miroslav Golijan, and Rui Du. Lossless data embedding for all image formats.

[14] Jiri Fridrich, Miroslav Golijan, and Rui Du. Invertible authentication watermark for jpeg images. In *Proc. Of the International Conference on Information Technology: Coding and Computing (ITCC'01)*, 2001.

[15] D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In *Proc. of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Pringciples of Database Systems(PODS 2003)*, 2003.

[16] E. Hildebrandt and S. Shenoi. User authentication in mulidatabases systems. In *Proc. 9th Int. Workshop on Database and Expert Systemes Applications*, pages 281–286, 1998.

[17] I.J.Cox and M.L.Miller. A review of watermarking and the importance of perceptual modeling. In *Proc. of Electronic Imaging*, February 1997.

[18] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proc. SIGMOD*, 1997.

[19] M.S. Kankanhalli, Rajmohan, and K.R. Ramakrishnan. Content-based watermarking of images. In *Proc. ACM International Conference on Multimedia (ACM MM 1998)*, 1998.

[20] S. Katzenbeisser and Editors F. Petitcolas. *Information Hiding Techniques for Steganography and Watermarking.* Artech House, Boston MA, 2000.

[21] S. Khanna and F. Zane. Watermarking maps: hiding information in structured data. In *Proc. Symposium on Discrete Algorithms (SODA)*, 2000.

[22] D. Kirovski, H. Malvar, and Y. Yacobi. Multimedia content screening using a dual watermarking and fingerprinting system. In *Proc. ACM International Conference on Multimedia (ACM MM 2002)*, 2002.

[23] J. Lach, W. H. Mangione-Smith, and M. Potkonjak. Enhanced intellectual property protection for digital circuits on programmable hardware. In *Proc. Information Hiding*, pages 286–301, 1999.

[24] Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases. In *Technical report, Center for Secure Information System, George Mason University, VA*, May 2003.

[25] MySQL Database System. *http://www.mysql.com.*

[26] M. Nyanchama and S. L. Osborn. Access rights administration in role-based security systems. In *Proc. IFIP Workshop on Database Security*, pages 37–56, 1994.

[27] J. Palsberg, S. Krishnaswamy, M. Kwon, Q. Shao D. Ma, and Y. Zhang. Experience with software watermarking. In *Proc. ACSAC 2000*, pages 308–316, 2000.

[28] J. J. K. Ó Ruanaidh and G. Csurka. A bayesian approach to spread spectrum watermark detection and secure copyright protection for digital image libraries. pages 207–212.

[29] R. Sion, M. Atallah, and S. Prabhakar. On watermarking numeric datasets. In *Proc. First International Workshop on Digital Watermarking (IWDW 2002)*, 2002.

[30] R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proc. ACM International Conference on Management of Data (SIGMOD 2003)*, 2003.

[31] W. Stallings. *Cryptographic Techniques and Data Security. Third Edition.* Prentice Hall, 2002.