Watermarking Relational Databases

Rakesh Agrawal **Jerry Kiernan**

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

Abstract

We enunciate the need for watermarking database relations to deter their piracy, identify the unique characteristics of relational data which pose new challenges for watermarking, and provide desirable properties of a watermarking system for relational data. A watermark can be applied to any database relation having attributes which are such that changes in a few of their values do not affect the applications.

We then present an effective watermarking technique geared for relational data. This technique ensures that some bit positions of some of the attributes of some of the tuples contain specific values. The tuples, attributes within a tuple, bit positions in an attribute, and specific bit values are all algorithmically determined under the control of a private key known only to the owner of the data. This bit pattern constitutes the watermark. Only if one has access to the private key can the watermark be detected with high probability. Detecting the watermark neither requires access to the original data nor the watermark. The watermark can be detected even in a small subset of a watermarked relation as long as the sample contains some of the marks.

Our extensive analysis shows that the proposed technique is robust against various forms of malicious attacks and updates to the data. Using an implementation running on DB2, we also show that the performance of the algorithms allows for their use in real world applications.

**Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002**

1 Introduction

The piracy of digital assets such as software, images, video, audio and text has long been a concern for owners of these assets. Protection of these assets is usually based upon the insertion of digital watermarks into the data [6] [11] [13]. The watermarking software introduces small errors into the object being watermarked. These intentional errors are called *marks* and all the marks together constitute the *watermark*. The marks must not have a significant impact on the usefulness of the data and they should be placed in such a way that a malicious user cannot destroy them without making the data less useful. Thus, watermarking does not prevent copying, but it deters illegal copying by providing a means for establishing the original ownership of a redistributed copy.

The increasing use of databases in applications beyond "behind-the-firewalls data processing" is creating a similar need for watermarking databases. For instance, in the semiconductor industry, parametric data on semiconductor parts is provided primarily by three companies: Aspect, IHS, and IC Master. They all employ a large number of people to manually extract part specifications from datasheets. They then license these databases at high prices to design engineers. Companies like Acxiom have compiled large collections of consumer and business data. In the life sciences industry, the primary assets of companies such as Celera are the databases of biological information. The Internet is exerting tremendous pressure on these data providers to create services (often referred to as e-utilities or web services) that allow users to search and access databases remotely. While this trend is a boon to end users, it is exposing the data providers to the threat of data theft. They are therefore demanding capabilities for identifying pirated copies of their data.

We suggest that rights management of relational data through watermarking should become an important topic for database research. Database relations that can be watermarked have attributes which are such that changes in a few values do not affect the applications. But are there real-world datasets that can tolerate a small amount of error without degrading their usability?

Consider the ACARS meteorological data used in building weather prediction models [3]. The wind vector and temperature accuracies in this data are estimated to be

within 1.8 m/s and 0.5° C respectively [3]. The errors introduced by watermarking can easily be constrained to lie within the measurement tolerance in this data. As another example, consider experimentally obtained gene expression datasets that are being analyzed using various data mining techniques [15]. The nature of some of the data sets and the analysis techniques is such that changes in a few data values will not affect the results. Similarly, the customer segmentation results of a consumer goods company will not be affected if the external provider of the supplementary data adds or subtracts some amount from a few transactions. Later in the paper, we report experimental results using a forest cover dataset. It contains measurements for variables such as elevation, aspect, slope, distance to hydrology and roadways, soil type, etc. Small changes in some of the measurements do not affect the usability of this data. Finally, consider the parametric data on semiconductor parts alluded to earlier. For many parameters, errors introduced by watermarking can be made to lie within the measurement tolerance. It is noteworthy that the publishers of books of mathematical tables (e.g. logarithm tables and astronomical ephemerides) have been introducing small errors in their tables for centuries to identify pirated copies [13].

1.1 Do We Need New Watermarking Techniques for Relational Data?

There is a rich body of literature on watermarking multimedia data [6] [11] [13]. Most of these techniques were initially developed for still images [12] and later extended to video [10] and audio sources [4] [8]. While there is much to learn from this literature, there are also new technical challenges due to the differences in the characteristics of relational and multimedia data. These differences include:

- A multimedia object consists of a large number of bits, with considerable redundancy. Thus, the watermark has a large cover in which to hide. A database relation consists of tuples, each of which represents a separate object. The watermark needs to be spread over these separate objects.
- The relative spatial/temporal positioning of various pieces of a multimedia object typically does not change. Tuples of a relation on the other hand constitute a set and there is no implied ordering between them.
- Portions of a multimedia object cannot be dropped or replaced arbitrarily without causing perceptual changes in the object. However, the pirate of a relation can simply drop some tuples or substitute them with tuples from other relations.

Because of these differences, techniques developed for multimedia data cannot be directly used for watermarking relations. To elaborate this point further, let us map a relation to an image by treating every attribute value as a pixel. Unfortunately, the "image" thus defined will lack many properties of a real image. For instance, pixels in a neighborhood in a real image are usually highly correlated and this assumption forms the basis of many techniques such as predictive coding for deciding watermark locations [9]. Several techniques first apply a transform (e.g. discrete Fourier, discrete cosine, Mellin-Fourier, Wavelet) to the image, insert the watermark in the transformed space, and then invert the transform [9]. The noise introduced by the watermarking signal is thus spread over the whole image. A direct application of these techniques to a relation will introduce errors in all of the attribute values, which might not be acceptable. Furthermore, such a watermark might not survive even minor updates to the relation.

Watermarking techniques for text exploit the special properties of formatted text. Watermarks are often introduced by altering the spacing between words and lines of text [17]. Some techniques rely on rephrasing some sentences in the text [1]. While these techniques might be useful to watermark relations containing CLOBs (character large binary objects), their applicability to relations consisting of simple data types is suspect.

Techniques for watermarking software have had limited success [5]. The problem is that the instructions in a computer program can often be rearranged without altering the semantics of the program. This resequencing can however destroy a watermark. Techniques have also been proposed to prevent copying of software. They however require installation of tamper resistant modules in users' machines, limiting their successful adoption in practice.

1.2 Our Contributions

We believe that the watermarking of relational data has significant technical challenges and practical applications to deserve serious attention from the database research community. A desiderata for a system for watermarking needs to be specified, followed by development of specific techniques. These techniques will most certainly use existing watermarking principles. However, they will also require enhancements to the current techniques as well as new innovations.

We have attempted to provide such a desiderata in this paper. To demonstrate the feasibility of watermarking relational data, we also present an effective technique that satisfies this desiderata. This technique marks only numeric attributes and assumes that the marked attributes can tolerate changes in some of the values. The basic idea is to ensure that some bit positions for some of the attributes of some of the tuples contain specific values. The tuples, attributes within a tuple, bit positions in an attribute, and specific bit values are all algorithmically determined under the control of a private key known only to the owner of the relation. This bit pattern constitutes the watermark. Only if one has access to the private key, can the watermark be detected with high probability. Our detailed analysis shows that the watermark can withstand a wide variety of malicious attacks.

1.3 Organization

The rest of the paper is organized as follows. Section 2 specifies our watermarking model and the desirable properties of a system for watermarking relational databases. Section 3 gives our algorithms for inserting and detecting watermarks. We also discuss its novelty with respect to existing work. Section 4 analyzes the properties of the proposed technique. Section 5 provides implementation details and an experimental evaluation. We conclude with a summary and directions for future work in Section 6.

2 Model

Say Alice is the owner of the relation R that contains η tuples, out of which she has marked ω tuples. The following properties are desirable.

Detectability Alice should be able to detect her watermark by examining ω tuples from a suspicious database. Clearly, if her bit pattern (watermark) is present in all of ω tuples, she has good reason to suspect piracy. However, it is reasonable for Alice to get suspicious if her pattern is present in at least τ tuples ($\tau \leq \omega$), where τ depends on ω and a preselected value α, called the significance level of the test. The value of τ is so determined that the probability that Alice will find by sheer chance her bit pattern in at least τ tuples out of ω tuples is less than α.

Robustness Watermarks should be robust against attacks to erase them. Say the attacker, Mallory[1], changes ζ tuples of Alice's relation R. We say that the watermark is safe from erasure if the attack is not able to destroy the marks of at least τ tuples, where τ depends as above on ω and α. We further discuss robustness in Section 2.1.

Incremental Updatability Having watermarked R, Alice should be able to update R in the future without destroying the watermark. As Alice adds/deletes tuples or modifies the values of attributes of R, the watermark should be incrementally updatable. That is, the watermark values should only be recomputed for the added or modified tuples.

Imperceptibility The modifications caused by marks should not reduce the usefulness of the database. In addition, the commonly used statistical measures such as mean and variance of the numerical attributes should not be significantly affected.

Blind System Watermark detection should neither require the knowledge of the original database nor the watermark. This property is critical as it allows the watermark to be detected in a copy of the database relation, irrespective of later updates to the original relation.

Key-Based System Following Kerckhoffs [14], the watermarking system should assume that the method used for inserting a watermark is public. Defense must lie only in the choice of the private key. The folly of "security-by-obscurity" has been shown repeatedly since the first enunciation of Kerckhoffs' principle in 1883 [13].

[1]The cryptography literature has conventionally given a male persona to Mallory, the malicious active attacker [18].

2.1 Benign Updates and Malicious Attacks

Since database relations are updatable, the marks contained in a relation can be removed by benign updates as well as malicious attacks.

Benign Updates Suppose Mallory has stolen Alice's data without realizing that it has been watermarked. Subsequently, Mallory may update the stolen data as he uses it. Watermarking should be such that Alice does not lose her watermark in the stolen data in spite of Mallory's updates.

Malicious Attacks Mallory may know that the data he has stolen contains a watermark, but he may try to erase the watermark or try other means for claiming false ownership. The watermarking system should protect Alice from various forms of Mallory's malicious attacks:

Bit Attacks The simplest malicious attack attempts to destroy the watermark by updating some bits. Clearly, if Mallory can change all the bits, he can easily destroy the watermark. However, he has also made his data completely useless. The effectiveness of an attack should therefore consider the relationship between the number of bits that Mallory and Alice change, since each change can be considered an error. Having more errors clearly makes the data less useful.

A *randomization* attack assigns random values to some number of bit positions. A *zero out* attack sets values of some number of bit positions to zero. A *bit flipping* attack inverts the values of some number of bit positions. Note that benign updates can also be modeled as a randomization attack.

Rounding Attack Mallory may try to lose the marks contained in a numeric attribute by rounding all the values of the attribute. This attack is not any better than the bit attacks discussed above. Mallory has to correctly guess how many bit positions are involved in the watermarking. If he underestimates it, his attack may not succeed. If he overestimates it, he has degraded the quality of his data more than necessary. Even if his guess is correct, his data will not be competitive against Alice's data because his data values are less precise.

A related attack will be one in which the numeric values are uniformly translated. For example, Mallory may translate using units of measurement (e.g., imperial units to metric units). Alice simply needs to convert the values back to the original system in order to recover the marks. In general, Mallory can apply arbitrary translations to numeric values. In this case, Mallory would also need to inform potential users of the conversion used which could also be applied by Alice before detecting her watermark. The unnecessary conversion would also raise suspicion among users.

Subset Attack Mallory may take a subset of the tuples or attributes of a watermarked relation and hope that the

η	Number of tuples in the relation
ν	Number of attributes in the relation available for marking
ξ	Number of least significant bits available for marking in an attribute
$1/\gamma$	Fraction of tuples marked
ω	Number of tuples marked
α	Significance level of the test for detecting a watermark
τ	Minimum number of correctly marked tuples needed for detection

Figure 1: Notation

watermark is lost.

Mix and Match Attack Mallory may create his relation by taking disjoint tuples from multiple relations containing similar information.

Additive Attack Mallory may simply add his watermark to Alice's watermarked relation and claim ownership.

Invertibility Attack Mallory may launch an invertibility attack [7] to claim ownership if he can successfully discover a fictitious watermark. Mallory's claimed watermark is in fact a random occurrence.

3 Algorithms

We now present a technique for watermarking database relations and show that it satisfies the desiderata outlined above. This technique marks only numeric attributes and assumes that the marked attributes are such that small changes in some of their values are acceptable and non-obvious. All of the numeric attributes of a relation need not be marked. The data owner is responsible for deciding which attributes are suitable for marking.

We are watermarking a database relation R whose scheme is $R(P, A_0, \ldots, A_{\nu-1})$, where P is the primary key attribute. (Section 3.5 gives extensions for watermarking a relation that does not have a primary key attribute.) For simplicity, assume that all ν attributes $A_0, \ldots, A_{\nu-1}$ are candidates for marking. They are all numeric attributes and their values are such that changes in ξ least significant bits for all of them are imperceptible.[2]

Gap γ is a control parameter that determines the number of tuples marked, $\omega \approx \eta/\gamma$. One can often trade-off γ against ξ that determines the extent of error introduced in an attribute's values. If less tuples are marked, it might be possible to introduce greater changes in the values of marked attributes.

We denote by $r.A_i$ the value of attribute A_i in tuple $r \in R$. Figure 1 summarizes the important parameters used in our algorithms. These algorithms make use of message authenticated codes that we briefly review next.

[2] It is not necessary to use consecutive ξ least significant bits for marking. For instance, we may not use those bit positions in which the distribution of bit values is skewed [16]. We omit this detail.

```
// The private key K is known only to the owner of the database.
// The parameters γ, ν, and ξ are also private to the owner.

1) foreach tuple r ∈ R do
2)     if (F(r.P) mod γ equals 0) then          // mark this tuple
3)         attribute_index i = F(r.P) mod ν     // mark attribute A_i
4)         bit_index j = F(r.P) mod ξ           // mark j^th bit
5)         r.A_i = mark(r.P, r.A_i, j)

6)  mark(primary_key pk, number v, bit_index j) return number

7)      first_hash = H(K ∘ pk)

8)      if (first_hash is even) then
9)          set the j^th least significant bit of v to 0
10)     else
11)         set the j^th least significant bit of v to 1

12)     return v
```

Figure 2: Watermark Insertion Algorithm

3.1 Message Authenticated Code

A one-way hash function $\mathcal{H}$ operates on an input message M of arbitrary length and returns a fixed length hash value h, i.e., $h = \mathcal{H}(M)$. It has the additional characteristics that i) given M, it is easy to compute h, ii) given h, it is hard to compute M such that $\mathcal{H}(M) = h$, and iii) given M, it is hard to find another message M' such that $\mathcal{H}(M) = \mathcal{H}(M')$. Several one-way functions have been described in [18]. MD5 and SHA are two good choices for $\mathcal{H}$.

A message authenticated code (MAC) is a one-way hash function that depends on a key. Let $\mathcal{F}$ be a MAC that randomizes the values of the primary key attribute $r.P$ of tuple r and returns an integer value in a wide range. $\mathcal{F}$ is seeded with a private key $\mathcal{K}$ known only to the owner. We use the following MAC, considered to be secure [18]:

$$\mathcal{F}(r.P) = \mathcal{H}(\mathcal{K} \circ \mathcal{H}(\mathcal{K} \circ r.P))$$

where $\circ$ represents concatenation.

3.2 Watermark Insertion

Figure 2 gives the watermark insertion algorithm[3]. Line 2 determines if the tuple under consideration will be marked. Because of the use of MAC, only the owner who has the knowledge of the private key $\mathcal{K}$ can easily determine which tuples have been marked. For a selected tuple, line 3 determines the attribute that will be marked amongst the ν candidate attributes. For a selected attribute, line 4 determines the bit position amongst ξ least significant bits that will be marked. Again, the results of the tests in lines 3 and 4 depend on the private key of the owner. For erasing a watermark, therefore, the attacker will have to guess not only the tuples, but also the marked attribute within a tuple as well as the bit position.

The mark subroutine sets the selected bit to 0 or 1 depending on the hash value obtained in line 7. Thus,

[3] The algorithm is written in a form that simplifies exposition, rather than in the most computationally efficient form.

```
// K, γ, ν, and ξ have the same values used for watermark insertion.
// α is the test significance level that the detector preselects.

1)  totalcount = matchcount = 0

2)  foreach tuple s ∈ S do
3)      if (F(s.P) mod γ equals 0) then        // this tuple was marked
4)          attribute_index i = F(s.P) mod ν   // attribute A_i was marked
5)          bit_index j = F(s.P) mod ξ         // j^th bit was marked
6)          totalcount = totalcount + 1
7)          matchcount = matchcount + match(s.P, s.A_i, j)

8)  τ = threshold(totalcount, α)               // see Section 4.2
9)  if (matchcount ≥ τ) then suspect piracy

10) match(primary_key pk, number v, bit_index j) return int

11)     first_hash = H(K ∘ pk)

12)     if (first_hash is even) then
13)         return 1 if the j^th least significant bit of v is 0 else return 0
14)     else
15)         return 1 if the j^th least significant bit of v is 1 else return 0
```

Figure 3: Watermark Detection Algorithm

the result of line 9 (line 11) either leaves the attribute value unchanged or decrements (increments) it. Consequently, marking decrements some of the values of an attribute while it increments some others and leaves some unchanged.

Databases usually allow attributes to assume null values. If a null attribute value is encountered while marking a tuple, we do not apply the mark to the null value, leaving it unchanged.

3.3 Watermark Detection

Assume Alice suspects that the relation S published by Mallory has been pirated from her relation R. The set of tuples and attributes in S can be a subset of R. We assume that Mallory does not drop the primary key attribute or change the value of primary keys since the primary key contains valuable information and changing it will render the database less useful from the user's point of view[4].

The watermark detection algorithm, shown in Figure 3, is probabilistic in nature. Line 3 determines if the tuple s under consideration must have been marked at the time of inserting the watermark. Lines 4 and 5 determine the attribute and the bit position that must have been marked. The subroutine match then compares the current bit value with the value that must have been set for that bit by the watermarking algorithm.

We thus know at Line 8 how many tuples were tested (totalcount) and how many of them contain the expected bit value (matchcount). In a probabilistic framework, only a certain minimum number of tuples have to contain matching marked bits. The matchcount is compared with the minimum count returned by the threshold function for the test to succeed at the chosen level of significance α. The threshold function is described in Section 4.2.

Figure 3 assumes for simplicity that all the candidate attributes $A_0, \ldots, A_{\nu-1}$ were present in S. If Alice finds a tuple s in which she must have marked the attribute A_i (line 4) but Mallory has omitted A_i, she simply ignores the tuple. Similarly, if a tuple is found whose attribute A_i should have been marked, but A_i has a null value, the tuple is ignored. I.e., The values of matchcount and totalcount are unaffected.

3.4 Remarks

Data Formats We rely on Java to handle issues related to data formats for numeric types. Java prescribes specific sizes for numeric types, which are machine independent. JVM also hides the complexities arising out of different byte orderings used on different machines for storing numeric data. Note that we mark the mantissa of a floating point number and decimal numbers are marked as integers ignoring scale.

Incremental Updatability Whether a tuple is marked or not depends on its primary key attribute. Thus a tuple can be inserted without examining the markings of any other tuple. Similarly, a tuple can be simply deleted. When updating the primary key attribute of a tuple, we recompute its marking before storing the tuple in the database. When updating a non-primary key attribute, nothing needs to be done if the algorithm has not selected this attribute for marking. On the other hand, if the attribute is a candidate for marking, the mark is applied to the attribute's value before storing it in the database.

Blind Watermarking The detection algorithm is blind. It simply extracts ω bits of information from the data, without requiring access to the original data or watermark to arrive at its decision. Blind watermarking is critical for database relations since relations are frequently updated. Each version of the relation would need to be kept if the original is required for detecting a watermark.

3.5 Relations Without Primary Keys

The watermarking technique described above is predicated on the existence of a primary key in the relation being watermarked. Primary keys arise naturally in real-life situations and we expect the majority of relations that someone would be interested in watermarking will have a primary key. We discuss next how to extend our technique if this assumption does not hold.

Assume first that the relation R consists of a single numeric attribute A. Partition the bits of the attribute A into two groups. χ bits of the value $r.A$ are used as the "primary key substitute" of the tuple r and the remaining ξ bits are used for marking. We can now use the scheme described

[4] If this assumption does not hold, use the technique in Section 3.5. If Mallory tries to make benign changes to the primary key values by transforming them into semantically equivalent values (e.g., uniformally change part numbers such as CY7C225A into CY-7C-225A), Alice can detect her watermark by first inverting the transformed primary key values.

earlier. This construction will work only if $r.A$ does not have many duplicates. Too many duplicates in χ bits of $r.A$ values will result in many identical marks which an attacker can exploit.

If the relation has more than one attribute, one of them can be used as the substitute and the remainder for marking. Choose the attribute that has minimum duplicates to serve as the substitute. The substitute can also be spread across more than one attribute to reduce duplicates. The drawback is that if one of these attributes is omitted by Mallory, Alice will not be able to detect the watermark.

3.6 Related Work

Dugley and Roche [9] classify the various techniques for watermarking images along the following dimensions: i) the method for selecting pixels where the watermark message will be hidden; ii) the choice of workspace to perform the hiding operation; iii) the strategy for formatting the message; iv) the method for merging the message and cover; and v) the operation needed for extracting the message. According to this framework, our technique uses a private key and the primary key to select the bit positions. We do the hiding in the original space. We do not have a fixed message; the bit pattern that constitutes the message is dynamically and algorithmically computed (and incrementally updated). The merging operation is a bit operation, driven by a truth table defined by the mark subroutine of the insertion algorithm. The extraction operation is the dual of the merge operation.

The closest to our technique in the Dugley-Roche space of image watermarking techniques is the patchwork algorithm [2]. This algorithm chooses random pairs of points (a_i, b_i) of an image in the spectral domain, and increases the brightness at a_i by 1 unit while correspondingly decreasing the brightness at b_i. Random changes to relational data can potentially introduce large errors. It is also not clear how to handle incremental updates and how to protect the watermark from various forms of attacks if one were to apply the patchwork algorithm to relational data.

Another closely related work is the technique proposed in [1] for watermarking a *sequence* of numbers. The basic idea is to modify the numbers, interpreted as integers, to force them to be quadratic residues or nonresidues modulo a secret prime, according to the parity of the next bit of a user-provided message. The watermark is repeated many times throughout the data. The advantage of our technique is that we do not require data to be ordered and hence our technique is robust in the presence of updates. We also do not have a fixed message that is encrypted and repeated in the data.

4 Analysis

We now analyze the properties of the proposed watermarking technique.

4.1 Cumulative Binomial Probability

Repeated independent trails are called Bernoulli trials if there are only two possible outcomes for each trial and their probabilities remain the same throughout the trials. Let $b(k; n, p)$ be the probability that n Bernoulli trials with probabilities p for success and $q = 1 - p$ for failure result in k successes and $n - k$ failures. Then

$$b(k; n, p) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k q^{n-k} \quad (1)$$

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!} \quad 0 \leq k \leq n \quad (2)$$

Denote the number of successes in n trials by $\mathbf{S}_n$. The probability of having at least k successes in n trials, the cumulative binomial probability, can be written as

$$\mathbf{P}\{\mathbf{S}_n \geq k\} = \sum_{i=k}^{n} b(i; n, p) \quad (3)$$

For brevity, define

$$B(k; n, p) := \sum_{i=k}^{n} b(i; n, p) \quad (4)$$

4.2 Probabilistic Framework

We can now specify the *threshold* subroutine used in Line 8 of Figure 3. Suppose totalcount = ω when Alice runs the detection algorithm. That is, Alice looks at ω bits and observes the number of bits whose values match those assigned by the marking algorithm. The probability that at least τ out of ω ramdom bits – each bit equal to 0 or 1 with equal probability, independent of the other bits – matches the assigned value is $B(\tau; \omega, 1/2)$. In other words, the probability that at least τ bits match by sheer chance is $B(\tau; \omega, 1/2)$. Therefore, the subroutine

subroutine threshold(ω, α) **return** count

returns minimum τ such that $B(\tau; \omega, 1/2) < \alpha$.

The significance level α determines how amenable the system is to false hits. That is, α is the probability that Alice will discover her watermark in a database relation not marked by her. By choosing lower values of α, Alice can increase her confidence that if the detection algorithm finds her watermark in a suspected relation, it probably is a pirated copy.

4.3 Detectability

We see from Section 4.2 that the detectability of a watermark depends on the significance level α and the number of marked tuples ω. The latter in turn depends on the number of tuples in the relation η and the gap parameter γ.

Watermark Detection Figure 4 plots the proportion of marked tuples that must have the correct watermark value

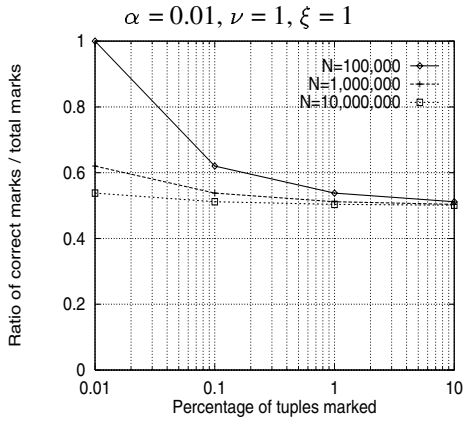

Figure 4: Proportion of correctly marked tuples needed for detectability

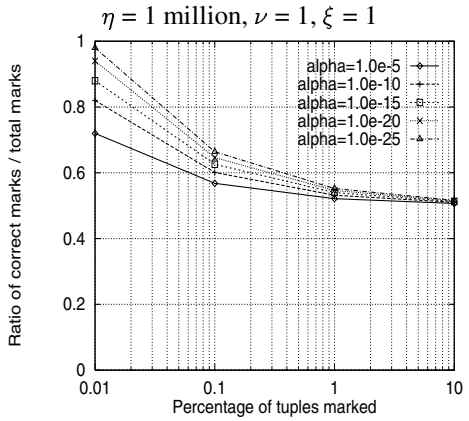

Figure 5: Proportion of correctly marked tuples needed for decreasing α (alpha $\equiv \alpha$)

for successful detection (i.e., τ/ω). We have plotted the results for relations of different sizes, assuming $\alpha = 0.01$. The X-axis varies the percentage of tuples marked (i.e. the fraction $1/\gamma$ expressed as percentage). The percentage of tuples marked 0.01%, 0.1%, 1.0%, and 10% correspond to the γ values of 10000, 1000, 100, and 10 respectively.

The figure shows that the required proportion of correctly marked tuples decreases as the percentage of marked tuples increases. This proportion also decreases as the number of tuples in the relation increases. We, of course, need more than 50% of the correctly marked tuples to differentiate a watermark from a chance occurrence, but with an appropriate choice of γ, this percentage can be made less than 51%. This figure also shows that for larger relations, we can mark a smaller percentage of the total number of tuples and yet maintain the detectability of the watermark.

In Figure 5, we have plotted the required proportion of correctly marked tuples for various values of α. The results are shown for a 1 million tuple relation. Clearly, we need to proportionately find a larger number of correctly marked tuples as the value of α decreases. More importantly though, even for very low values of α, it is possible to detect the watermark. Even for $\gamma = 10000$ where there are only 100 marked tuples out of 1 million tuples, it is possible to detect the watermark if 82% of the marks have the correct value for α as low as $1.0e-10$. For $\gamma = 10$, less than 52% of the marked tuples are required to have the correct mark for all α values.

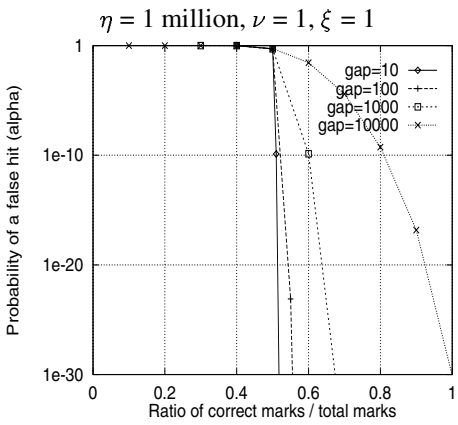

Figure 6: Probability of recovering by chance τ tuples having the correct watermark value out of ω tuples (gap $\equiv \gamma$)

False Hits Figure 6 illustrates the robustness of the system against false hits. The graph has been plotted for a 1 million tuple relation and for different values of the gap parameter γ. We have varied on the X-axis the ratio of marked tuples τ having the correct watermark value to the total number of marked tuples ω. The Y-axis shows $B(\tau; \omega, 1/2)$, which is the probability of falsely finding at least τ tuples having the correct watermark value out of ω tuples.

The graph shows that there is a sharp decrease in the probability that, by chance alone, more than 50% of the tuples will have a correct mark. For $\gamma = 10$, the probability that 51% or more of the marked tuples have a correct mark by chance is 1.29e-10. Even for a very large $\gamma = 10000$, the probability that 80% or more of the marked tuples have a correct mark by chance is only 5.58e-10. Thus, by choosing appropriate values for γ and α, false hits can be made highly improbable.

4.4 Robustness

We now analyze the robustness of our watermarking technique against various forms of malicious attacks. Alice has marked $\omega (\approx \eta/\gamma)$ tuples. For detecting her watermark, she uses the significance level of α that determines the minimum number of tuples τ out of ω that must have her mark intact.

4.4.1 Bit-Flipping Attack

In this form of attack, Mallory tries to destroy Alice's watermark by flipping the value at the bit positions he guesses have been marked. The analysis and results are similar for the zero-out and randomization attacks.

Assume that Mallory magically knows the values of the ν and ξ parameters used by Alice. The value of ξ is assumed to be the same for all of ν attributes. Since Mallory does not know which bit positions have been marked, he

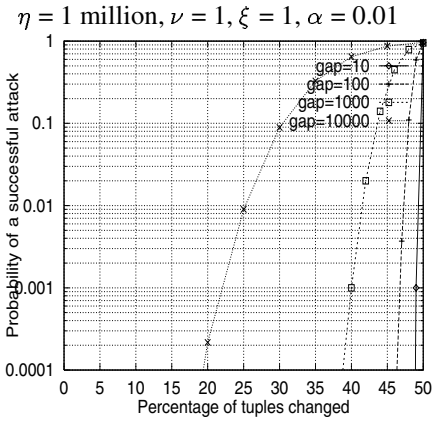

Figure 7: Probability of a successful attack (gap $\equiv \gamma$)

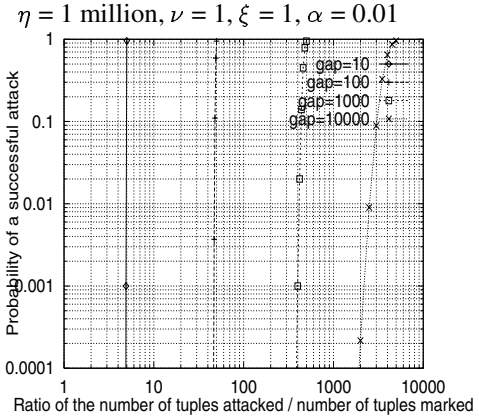

Figure 8: Excess error in destroying a watermark (gap $\equiv \gamma$)

randomly chooses ζ tuples out of η tuples. For every selected tuple, he flips all of the bits in all of ξ bit positions in all of ν attributes. To be successful, he should be able to flip at least $\bar{\tau} = \omega - \tau + 1$ marks.

The probability that this attack will succeed can be estimated as

$$\sum_{i=\bar{\tau}}^{\omega} \frac{\binom{\omega}{i}\binom{\eta-\omega}{\zeta-i}}{\binom{\eta}{\zeta}} \tag{5}$$

Essentially, Mallory is sampling without replacement ζ tuples out of η potentially marked tuples, hoping that at least $\bar{\tau}$ tuples out of ω marked tuples will show up in his sample.

Probability of Success Figure 7 shows the probability of success of the above attack on a 1 million tuple relation with α set to 0.01. We have assumed that Alice has marked only the least significant bit of one attribute and this information is somehow known to Mallory. We have varied on the X-axis the percentage of tuples changed. For $\gamma = 10,000$, if Mallory changes 40% of the tuples, he has 64% chance of destroying the watermark. For $\gamma = 1000$, he has to change 46% of the tuples to get 44% chance of success. His chance of success is only 11% if he changes

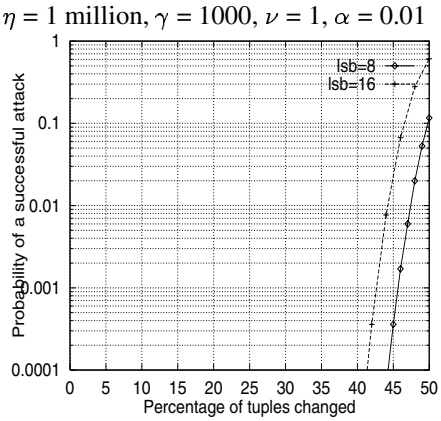

Figure 9: Probability of a successful attack when (lsb $\equiv \xi$) is underestimated by 1

48% of the tuples when $\gamma = 100$. Anything less than close to 50% does not destroy the watermark for $\gamma = 10$. Note that it is not in Mallory's interest to flip more than 50% of the marked bits. For in that case, Alice can detect the watermark by reflipping the appropriate bits and applying the detection algorithm on the transformed data.

Figure 8 shows the excess error introduced by Mallory in destroying a watermark, expressed as the ratio of the total number of tuples attacked to the number of marked tuples. When $\gamma = 10000$, Mallory has to change nearly 5000 times the number of tuples that were marked to destroy the watermark. Thus, Mallory's data will contain more than 3 orders of magnitude of error than Alice's version. This ratio becomes roughly 500, 50, and 5 respectively for γ = 1000, 100, and 10. Thus, Alice can choose a value of γ depending upon how tolerant the data is to errors and force Mallory to commit much larger errors (and hence make Mallory's data less desirable).

Figure 8 also says that if Mallory were restricted to making at most as many changes to the data as Alice, he could not have destroyed Alice's watermark. Hence, if the data is such that there is a limit to the number of changes it can tolerate without rendering it useless and Alice introduces the maximum possible number of changes, Mallory will not be able to remove the watermark.

Varying ξ Now consider the case where Alice marks one of $\xi > 1$ least significant bits, but only in one attribute. First assume Mallory somehow knows the exact value of ξ. However, Mallory will not know which bit position has been marked in a specific tuple. Having decided on a tuple, he will therefore have to flip all of ξ least significant bits of the attribute in order to destroy a mark. If we plot the probability of a successful attack against the percentage of tuples changed under this scenario, we get a graph identical to Figure 7. The difference is that Mallory has ξ times excess error. Mallory has to change ξ bits in an attribute value whereas Alice continues to change only one bit. A larger value of ξ also leads to larger errors in Mallory's data.

Suppose now that Mallory does not know the exact value of ξ used by Alice. Figure 9 shows what happens when

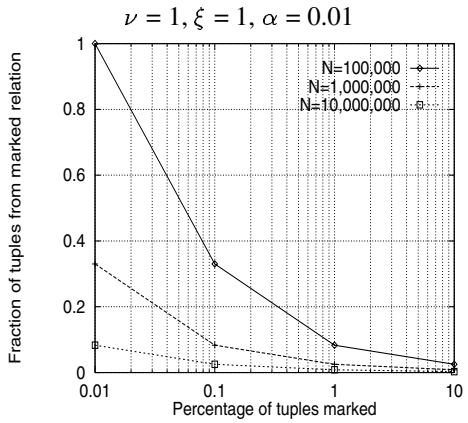

Figure 10: Minimum fraction of tuples from the watermarked relation needed for detectability

Mallory underestimates the value of ξ by just 1 bit. We take η = 1 million, γ = 1000, $\nu = 1$, and $\alpha = 0.01$, and plot the probability of a successful attack for different values of ξ and the percentage of tuples changed. Contrast Figure 9 with Figure 7 for the case whence $\gamma = 1000$. Compared to Figure 7, plots shift to the right showing a reduction in the probability of a successful attack. In fact, Figure 9 contains no values for $\xi = 4$ and 2. This is because the probability of success has become very small (less than 0.0001).

In reality, it is hard for Mallory to guess the exact value of ξ used by Alice. If he overestimates the value, he ends up introducing large errors; if he underestimates it, his chances of success are reduced. Thus, Alice can effectively use the ξ parameter to foil Mallory.

Varying ν Alice has the additional flexibility of marking any one of the ν attributes. Mallory, unfortunately, has to change the values of all of ν attributes in a tuple. The analysis of having ν attributes to mark is identical to having ν least significant bits to mark. Clearly, Alice should use all the attributes she can mark without affecting the quality of data significantly.

4.4.2 Mix-and-Match Attack

In the mix-and-match attack, Mallory takes κ fraction of tuples from Alice's relation R and mixes them with tuples from other sources to create his relation S of the same size as R. We give a simple average case analysis for this attack. For Alice to be able to detect her watermark in S, we need

$$\kappa\frac{\eta}{\gamma} + \frac{1}{2}(1-\kappa)\frac{\eta}{\gamma} \;\geq\; \tau \qquad (6)$$

The second term on the left hand side of the inequality arises because the detection algorithm, when applied to an unmarked relation, can expect to find matching bits in half of the tuples. The value of τ depends on η, γ, and α.

Figure 10 gives the minimum value of κ for which Alice can still detect the watermark for various values of η and γ (expressed as the percentage of tuples marked). We have taken α to be 0.01 and $\nu = \xi = 1$ and extracted τ values from Figure 4. We see that if Alice had marked 10% of the tuples, she can detect her watermark even when Mallory pirates less than 2.5% of tuples from her 100,000 tuples relation. The fraction of tuples Mallory can pirate and yet avoid detection comes down rapidly as the size of Alice's relation increases.

4.4.3 Additive Attack

In the additive attack, Mallory simply inserts his watermark in Alice's data. The original ownership claim can be resolved by locating the overlapping regions of the two watermarks in which the bit values of the marks conflict and determining which owner's marks win. The winner must have overwritten the loser's bits and hence must have inserted the watermark later. Depending upon the significance level chosen for the test, it is possible not to reach a decision if only a few marks collide. Clearly having more marked tuples (i.e. smaller value of γ) increases collisions and hence the chance of reaching a decision.

4.4.4 Invertibility Attack

Counterfeit watermarks are used to claim ownership in an invertibility attack [7]. This attack translates into Mallory being able to find a key that yields a satisfactory watermark for some value of α. The key discovered by Mallory need not match the key used by Alice in inserting her watermark. For high values of α, Mallory can stumble upon such a key by repeatedly trying different key values. This attack is thwarted by using low values of α (e.g., 1.0e-10), rendering negligible the probability of accidentally finding a good key.

4.5 Design Trade-Offs

Our watermarking technique has four important tunable parameters: i) α, the test significance level, ii) γ, the gap parameter that determines the fraction of tuples marked, iii) ν, the number of attributes in the relation available for marking, and iv) ξ, the number of least significant bits available for marking. Based on the analysis presented in this section, we have summarized in Figure 11 the important trade-offs when selecting the values for these parameters.

$\downarrow \alpha$	$\downarrow$ false hits	$\uparrow$ missed watermarks
$\downarrow \gamma$	$\uparrow$ robustness	$\uparrow$ data errors
$\uparrow \nu$	$\uparrow$ robustness	
$\uparrow \xi$	$\uparrow$ robustness	$\uparrow$ data errors

Figure 11: Design Trade-Offs

5 Implementation and Experiments

We next provide some implementation notes and experimental results obtained using a real-life dataset.

5.1 Implementation

We describe an implementation in which watermark insertion as well as detection are implemented as database client

programs using SQL-92 level capabilities. We assume for simplicity that the primary key of the relation R consists of a single attribute P and only one attribute A is available for watermarking.

A watermark is inserted by first retrieving tuples of R, with attributes P, A specified in the select list. The select statement contains the additional clause "for update of A" that lets the database engine know that the selected tuples of R will be updated. For each tuple r thus fetched, if the watermarking algorithm determines that a change is needed in the value of $r.A$, an update statement is issued to mark $r.A$. The update statement has a "current of cursor" clause that lets the database engine know that the tuple to be updated is r.

Watermark detection is performed using a select statement to fetch the tuples of the suspicious database relation S, specifying the attributes P, A in the select list. Appropriate counts for *totalcount* and *matchcount* are incremented for every result tuple. Finally, if the probability of finding *matchcount* marks in *totalcount* tuples is within the significance level, the watermark has been detected.

5.2 Experimental Results

We now report some experimental results that complement the analysis presented in Section 4. Experiments were performed using the Forest Cover Type dataset, available from the University of California–Irvine KDD Archive (kdd.ics.uci.edu/databases/covertype/covertype.html). The dataset has 581,012 rows, each with 61 attributes. We added an extra attribute called *id* to serve as the primary key. We chose the first ten integer-valued attributes as candidates for watermarking.

We ran experiments on DB2 UDB Version 7 using JDBC connectivity on a Windows NT Version 4.00 workstation with a 400MHz Intel processor, 128 MB of memory, and a 10 GB disk drive. We used the default DB2 settings, except for the log file and the lock list. It was necessary to modify these settings because some of our experiments were update intensive. The log file size was set to 20 MB and the lock list to 2 MB.

5.2.1 Watermarking Overhead

We ran two experiments to assess the computational cost of watermarking and detection. Performance was measured in elapsed time. Each experiment was repeated 30 times and the overhead ratios were computed from the summation of individual trials.

The first experiment evaluated the cost of inserting a watermark. We tried the worst case by setting γ to 1. In this case, the watermarking algorithm will read η tuples and find that every tuple requires marking. However, on average, half the tuples will already have the correct value for the mark. Therefore, we expect that watermarking will update only $\eta/2$ tuples. We compare these latencies to the time required to read η tuples and update $\eta/2$ tuples. The comparison yielded a ratio of 1.16, showing a rather small overhead of 16% incurred by watermarking. This overhead is due to the cost of computing hash values needed to determine the mark for individual tuples. The average elapsed time to watermark the relation was 2245 seconds (roughly 37 minutes). This time included the cost of logging updates to half the tuples in the relation.

The second experiment assessed the cost of detection. We again chose the worst case by setting γ to 1 and by choosing the sample size for detecting the watermark to be the entire relation. The experiment compared the time required to detect η marks across η tuples against the time required to simply read η tuples. The comparison yielded a ratio of 4.38. If this cost seems high, we should point out that DB2 has very good sequential read performance due to smart prefetching. The major component of the cost of detection is the computation of one way hash functions needed to determine the presence of the mark for each tuple. The average detection time was only 214 seconds (roughly 4 minutes).

These results indicate that our algorithms have adequate performance to allow for their use in real world applications.

5.2.2 Imperceptibility

We next report the impact of watermarking on the mean and variance of values of marked attributes. This experiment was done by varying γ from 10 to 10000 and by varying ξ from 1 to 8. We found a minuscule change in the mean value for all the attributes. Table 1 shows changes in variance for different attributes. The values have been rounded to the nearest integer. An empty entry indicates very little or no change. As expected, greater changes in variance occur when ξ is large and γ is small because of larger perturbations in a greater fraction of tuples. Overall, the changes are insignificant given the amount of original variance. The only significant change occurred in the *Slope* attribute for $\xi = 8, \gamma = 10$. Compared to other attributes, this attribute has relatively small values that are perturbed significantly when ξ is large. Note that if these changes seem significant, ν, ξ and γ parameters can be adjusted to reduce the impact of watermarking on the data.

These results can be understood as follows. When an attribute value is marked, there is 1/2 probability that the value will not change. A bit with value 1 is converted to 0 with probability 1/4 and vice versa. Thus, an original value v will remain v with probability 1/2 and will become $v + \epsilon$ or $v - \epsilon$, each with probability 1/4. Hence, if every value of an attribute is equally likely to be selected and it is as likely that the value will be incremented as decremented then the mean and variance will not be affected significantly.

5.2.3 Detectability in the Presence of Subset Attacks

This set of experiments study the impact on the detectability of a watermark if only a subset of the watermarked relation is available for detection. We considered several levels of tuple selectivity σ which determines the percentage of tuples retrieved from the relation. For each selectivity

			$\gamma = 10000$			1000			100			10		
Attribute	**Mean**	**Variance**	$\xi = 1$	4	8	1	4	8	1	4	8	1	4	8
Elevation	2959	78391									+3			+16
Aspect	155	12525									+1			+8
Slope	14	56									+1			+14
Horz-Dist-To-Hydrology	269	45177									+1		+1	
Vert-Dist-To-Hydrology	46	3398						+1			+2			+11
Horz-Dist-To-Roadways	2350	2431272			-1		-1	-2			-9			+6
Hillshade-9am	212	717									+1			+12
Hillshade-Noon	223	391									+1			+12
Hillshade-3pm	142	1465									+1			+10
Horz-Dist-To-Fire-Points	1980	1753490						+1			-3			-1

Table 1: Change in variance introduced by watermarking

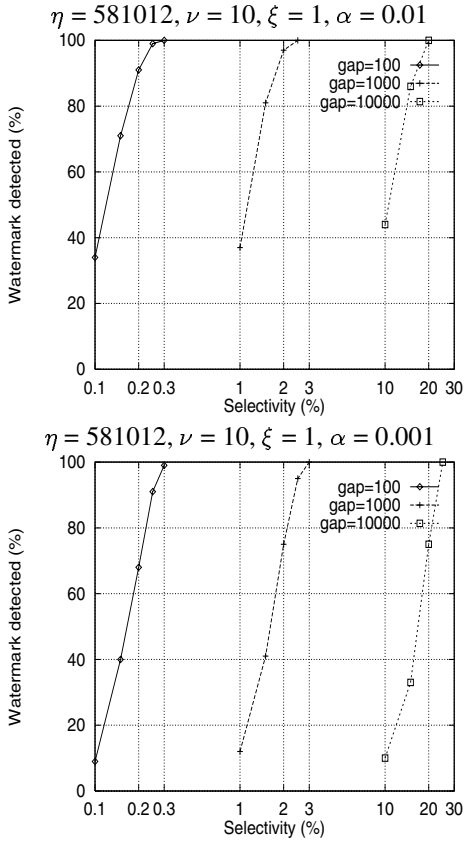

Figure 12: Percentage of samples in which the watermarks could be detected (gap $\equiv \gamma$)

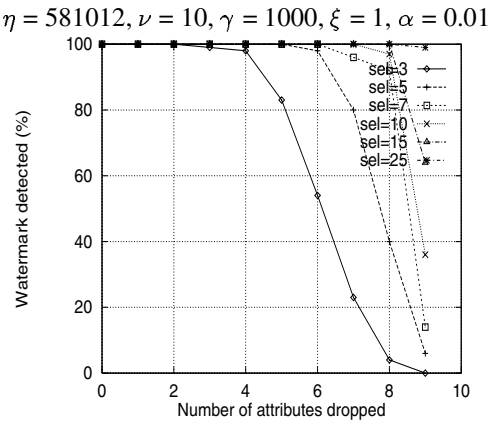

Figure 13: Percentage of samples in which the watermark could be detected when some of the watermarked attributes are dropped (sel $\equiv \sigma$)

level, we took 100 random samples and computed the percentage of samples in which the watermark could be found. The experiments were performed for γ values of 10, 100, 1000, and 10000.

Figure 12 shows the results for significance level $\alpha = 0.01$ and 0.001. We see that a watermark can be detected even if a large fraction of tuples have been omitted from the original data. For $\gamma = 10$, the watermark was detected in 100% of the samples for all selectivities. As expected, when γ increases (i.e. less tuples are marked), we need a higher percentage for selectivity (i.e. a larger percentage of tuples from the relation) to be able to detect the watermark. A slightly bigger sample is needed for smaller α for the same rate of success in detecting the watermark. However, a large reduction in the value of α (hence greater confidence in the test) does not require a large increase in the minimum size of the sample needed for detecting the watermark.

Figure 13 shows the effect of omitting watermarked attributes from the database sample. We plot the results for $\gamma = 1000$ and $\alpha = 0.01$, and vary the number of attributes dropped. Note from Figure 12 that the watermark could be detected in 100% of the samples for $\sigma = 3\%$. To study the effect of dropping marked attributes in isolation, therefore, we started with $\sigma = 3\%$ and tried larger values. For each selectivity and the number of attributes dropped, we took 100 random samples and computed the percentage of samples in which the watermark could be detected. The desired number of attributes are randomly selected for inclusion in a sample. The figure shows that we could drop 4 out of 10 marked attributes without losing detectability. As the number of attributes dropped increases, we need a larger sample to maintain detectability. Even when 9 out of 10 marked attributes were dropped, the watermark could be detected in 99% of the trials if the sample contained 25% of the marked relation.

These results can be analyzed as follows. Let us represent by π the fraction of marked attributes included in the sample. Since the probability of finding a matching marked bit by chance is 1/2, we need to be able to find at least 7 correct marks to detect a watermark at significance level $\alpha = 0.01$. This number increases to 10 for $\alpha = 0.001$. Let us represent this number by k. That is, if a sample con-

tains k marked tuples, we can detect the watermark at the desired significance level (assuming the watermark has not been corrupted). If the marked tuples were uniformly distributed and marks were uniformly distributed amongst the candidate attributes and we could get a truly random sample, it will be sufficient to have a sample obtained by using a selectivity σ such that $\pi\sigma\eta/\gamma \geq k$. In the absence of this fortunate situation, a rule of thumb will be to choose σ such that $\pi\sigma\eta/\gamma \geq 2k$.

The graphs in Figure 12 and 13 exhibit this behavior. Consider for instance the case of $\gamma = 1000$ and $\alpha = 0.01$ in Figure 12. Since $\eta = 581012$, $k = 7$ and $\pi = 10/10$, we need a minimum σ of 2.5% for 100% watermark detection. For smaller values of σ, the percentage of samples in which watermark is detected comes down.

These experiments show that our watermark detection algorithm is robust even when an attacker drops some of the tuples or the watermarked attributes from the relation. Moreover, depending upon the number of attributes omitted and the number of tuples dropped, we can estimate the size of the sample needed for detecting the watermark.

6 Summary

The following are the major contributions of this paper:

- Identification of the rights management of relational data through watermarking as an important and technically challenging problem for database research.
- Articulation of the desirable properties of a watermarking system for relational data.
- Enunciation of the various forms of malicious attacks from which the watermark inserted in a relation must be protected.
- First proposal of a watermarking technique specifically geared for relational data.
- Extensive analysis and empirical evaluation of the robustness and effectiveness of the proposed technique to demonstrate the feasibility of watermarking real-life datasets.

In the future, we would like to extend the proposed watermarking technique to also mark non-numeric attributes. We also plan to address the related problem of fingerprinting [13] [19] to be able to identify the culprit in cases where there can be multiple sources of piracy.

Acknowledgments We wish to thank Alexandre Evfimievsky for providing the code for computing cumulative binomial probability for large values. We are also thankful to Ramesh Agarwal, Dan Gruhl and Peter Haas for their thoughtful comments.

References

[1] M. Atallah and S. Wagstaff. Watermarking with quadratic residues. In *Proc. of IS&T/SPIE Conference on Security and Watermarking of Multimedia Contents*, January 1999.

[2] W. Bender, D. Gruhl, and N. Morimoto. Techniques for data hiding. In *Proc. of the SPIE 2420 (Storage and Retrieval for Image and Video Databases III)*, pages 164–173, 1995.

[3] S. Benjamin, B. Schwartz, and R. Cole. Accuracy of ACARS wind and temperature observations determined by collocation. *Weather and Forecasting*, 14:1032–1038, 1999.

[4] L. Boney, A. H. Tewfik, and K. N. Hamdy. Digital watermarks for audio signals. In *International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996.

[5] C. S. Collberg and C. Thomborson. Watermarking, Tamper-Proofing, and Obfuscation — Tools for Software Protection. Technical Report 2000-03, University of Arizona, Feb 2000.

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

[7] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal of Selected Areas in Communications*, 16(4):573–586, 1998.

[8] S. Czerwinski. Digital music distribution and audio watermarking. Available from `http://citeseer.nj.nec.com`.

[9] J.-L. Dugelay and S. Roche. A survey of current watermarking techniques. In S. Katzenbeisser and F. A. Petitcolas, editors, *Information Hiding Techniques for Steganography and Digital Watermarking*, chapter 6, pages 121–148. Artech House, 2000.

[10] F. Hartung and B. Girod. Watermarking of uncompressed and compressed video. *Signal Processing*, 66(3):283–301, 1998.

[11] N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Steganography and Watermarking – Attacks and Countermeasures*. Kluwer Academic Publishers, 2000.

[12] Joseph J. K. Ó Ruanaidh, W. J. Dowling, and F. M. Boland. Watermarking digital images for copyright protection. *IEEE Proceedings on Vision, Signal and Image Processing*, 143(4):250–256, 1996.

[13] S. Katzenbeisser and F. A. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[14] A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9:5–38, January 1883.

[15] E. Lander. Array of hope. *Nature Genetics*, 21:3–4, 1999.

[16] M. Maes. Twin peaks: The histogram attack on fixed depth image watermarks. In *Proc. of the 2nd International Workshop on Information Hiding*, pages 290–305. Springer-Verlag Lecture Notes in Computer Science 1525, 1998.

[17] N. Maxemchuk. Electronic document distribution. Technical Journal, AT&T Labs, September 1994.

[18] B. Schneier. *Applied Cryptography*. John Wiley, second edition, 1996.

[19] N. R. Wagner. Fingerprinting. In *IEEE Symp. on Security and Privacy*, pages 18–22, Oakland, California, April 1983.