Watermarking relational data: framework, algorithms and analysis[*]

Rakesh Agrawal, Peter J. Haas, Jerry Kiernan

IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA;
e-mail: ragrawal@acm.org

Abstract. We enunciate the need for watermarking database relations to deter data piracy, identify the characteristics of relational data that pose unique challenges for watermarking, and delineate desirable properties of a watermarking system for relational data. 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 specific bit locations and values are algorithmically determined under the control of a secret key known only to the owner of the data. This bit pattern constitutes the watermark. Only if one has access to the secret key can the watermark be detected with high probability. Detecting the watermark requires access neither to the original data nor the watermark, and the watermark can be easily and efficiently maintained in the presence of insertions, updates, and deletions. Our analysis shows that the proposed technique is robust against various forms of malicious attacks as well as benign updates to the data. Using an implementation running on DB2, we also show that the algorithms perform well enough to be used in real-world applications.

Keywords: Watermarking – Steganography – Information hiding

1 Introduction

The piracy of software, images, video, audio, and text has long been a concern for owners of these digital assets. Protection schemes are usually based upon the insertion of digital watermarks into the data [5,10,12]. 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 are chosen so as to have an insignificant impact on the usefulness of the data and are placed in such a way that a malicious user cannot destroy them without making the data significantly less useful. Although watermarking does not prevent illegal copying, it deters such 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. The Internet is exerting tremendous pressure on data providers to create services that allow users to search and access databases remotely. Although this trend is a boon to end users, it exposes the data providers to the threat of data theft. Providers are therefore demanding technology for identifying pirated copies of their databases.

1.1 The need for new watermarking techniques

There is a rich body of literature on watermarking multimedia data [5,10,12]. Most of these techniques were initially developed for still images [11] and later extended to video [9] and audio sources [3]. While there is much to learn from this literature, there are also new technical challenges because relational and multimedia data differ in a number of important respects:

- 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.
- Multimedia objects typically remain intact; portions of an object cannot be dropped or replaced arbitrarily without causing perceptual changes in the object. On the other hand, tuple insertions, deletions, and updates are the norm in the database setting.

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

---

[*] A preliminary version of this paper appeared in the Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

forms the basis of many techniques such as predictive coding for deciding watermark locations [8]. 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 [8]. 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.

Similarly, watermarking techniques for text are not applicable to databases because these techniques typically exploit special properties of text formatting and semantics. For example, watermarks are often introduced by altering the spacing between words and lines of text [17]. Other techniques rely on rephrasing some sentences in the text [1]. Although these techniques might be useful for watermarking relations containing CLOBs (character large binary objects), their applicability to relations consisting of simple data types seems dubious.

Techniques for watermarking software not only seem to hold little promise for database applications but have had only limited success in their native domain [4]. A key problem is that the instructions in a computer program can often be rearranged without altering the semantics of the program. This resequencing can destroy a watermark. Techniques have also been proposed to prevent copying of software, but they require installation of tamper-resistant modules in users' machines, which limits their successful adoption in practice.

1.2 Our contributions

The watermarking of relational data has significant technical challenges and practical applications that deserve serious research effort. Desiderata for database watermarking systems need to be specified, followed by development of specific techniques. These techniques will almost certainly incorporate existing watermarking principles but will also require enhancements and innovations.

In this paper, we provide a set of desiderata and, to demonstrate the feasibility of watermarking relational data, propose an effective technique that satisfies these desiderata. Our technique marks numeric attributes. Specifically, we 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 secret key known only to the owner of the relation. This bit pattern constitutes the watermark. Only if one has access to the secret key can the watermark be detected with high probability. Our analysis shows that the watermark can withstand a wide variety of malicious attacks as well as benign updates.

1.3 Assumptions

A basic principle followed in the literature on watermarking multimedia content is that the watermark should be hidden in an imperceptible manner in the perceptive (meaningful) part of the content [5]. In the database setting, this principle requires insertion of a watermark into those tuples and attributes whose

omission will significantly decrease the value of the data. For simplicity, we assume throughout that every tuple of the relation is important and the owner is responsible for identifying attributes that define the perceptive part of the relation.

Suppose that the perceptive part of the relation consists of a set $\mathcal{A}$ of attributes. We make the following fundamental assumption:

> The relational table being watermarked is such that if all (or a large number of) ξ least significant bits of any attribute $A \in \mathcal{A}$ are dropped or perturbed, then the value of data is significantly reduced. However, it is possible to change a small number of the bits and not decrease the value of the data significantly.

One could argue that data with errors, however small, are not as valuable as data without any errors. However, we assume that the decrease in the value of the data is small enough that the owner is willing to pay this price in exchange for the ability to assert ownership.

Examples of real-world data sets that satisfy the above assumption include tables of parametric specifications (mechanical, electrical, electronic, chemical, etc.), surveys (geological, climatic, etc.), and life sciences data (e.g., gene expression). It is noteworthy that the publishers of books of mathematical tables – such as logarithm tables and astronomical ephemerides – have been introducing small errors in their tables for centuries to identify pirated copies [12].

1.4 Organization

The rest of the paper is organized as follows. Section 2 describes the challenges facing the designer of a watermarking system: we first outline various processes that can damage, erase, or otherwise compromise a watermark and then specify the most important requirements that a watermarking system must satisfy. Section 3 gives our algorithms for inserting and detecting watermarks. We also discuss the novelty of these algorithms with respect to existing work. Sections 4 and 5 analyze the properties of the proposed technique, and Sect. 6 provides an experimental evaluation. We conclude with a summary and directions for future work in Sect. 7.

2 Challenges for watermarking

There are many ways in which a watermark can potentially be damaged, erased, or compromised. A watermarking system must be resistant to both intentional and unintentional assaults, while not hindering ordinary data-processing operations.

2.1 Threats to a watermark

Suppose that Alice is the owner of a relation R that contains η tuples, out of which she has marked ω tuples. Activities that

can damage, erase, or compromise Alice's watermark include the following.

Benign updates   In the course of ordinary data processing, marked rows may be deleted and/or a small number of marked bits may erroneously be flipped.

Malicious attacks   An attacker, Mallory,[1] may steal the data and try to erase the watermark. Mallory's malicious attack may take various forms:

- *Bit Attacks:* The simplest malicious attack attempts to destroy the watermark by altering one or more bits, e.g., by deterministically flipping each bit or by setting each bit to 0 or 1 according to the independent toss of a fair coin. The effectiveness of such an attack is sensitive to the number of altered bits: if Mallory alters every bit in the database, then he can easily destroy the watermark, but he has also made his data completely useless. The more information Mallory has about the locations of the marked bits, the more effective his attack.
- *Rounding Attack:* Mallory may try to lose the marks contained in a numeric attribute by rounding all the values of the attribute. For this attack to be effective, Mallory needs to correctly guess the number of bit positions involved in the watermark. If he underestimates this number, 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 be inferior to Alice's data because his data values are less precise.
- *Transformations:* An attack related to the rounding attack is one in which the numeric values are linearly transformed. For example, Mallory may convert the data to a different unit of measurement (e.g., Fahrenheit to Celsius). Alice simply needs to convert the values back to the original system in order to recover the marks. In general, Mallory can apply arbitrary transformations to numeric values. In this case, Mallory would also need to inform potential users of the transformation used, and Alice could then invert the transformation before detecting her watermark. Any blatantly unnecessary conversion by Mallory would likely raise suspicion among users.
- *Subset Attack:* Mallory may take a subset of the tuples or attributes of a watermarked relation and hope that the watermark is lost.
- *Mix-and-Match Attack:* Mallory may create his relation by taking disjoint tuples from multiple relations containing similar information.

False claims of ownership   Mallory may try to establish a plausible (but false) claim of ownership. For example, Mallory may simply add his watermark to Alice's watermarked relation and claim ownership. In a related scheme, Mallory may launch an "invertibility attack" [7] to claim ownership if he can successfully discover a spurious watermark by trial and error.

2.2 Desiderata for a watermarking system

A watermarking system should satisfy the following properties:

Robustness   Watermarks should be robust against degradation caused by either benign updates or malicious attacks, as described in Sect. 2.1.

Accuracy   Alice should, with high probability, not detect her watermark in someone else's nonpirated database. We call such an erroneous detection a *false hit*.

Incremental updatability   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.

Blind system   Watermark detection should not require the knowledge of either the original database or 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.

Public system   Following Kerckhoffs [13], the watermarking system should assume that the method used for inserting a watermark is public. Defense must lie only in the choice of the secret key. The folly of "security by obscurity" has been shown repeatedly since the first enunciation of Kerckhoffs' principle in 1883 [12].

3 Algorithms

We now present a technique for watermarking database relations that satisfies the desiderata given above. Our technique marks only numeric attributes and assumes that the marked attributes are such that small changes in some of their values are acceptable and nonobvious. 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.

Suppose that 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. (Sect. 3.6 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. Thus each attribute is numeric with values such that small changes in ξ least significant bits are imperceptible.[2]

The *gap* γ is a control parameter that determines the number ω of tuples marked via the approximate relationship $\omega \approx \eta/\gamma$. One can often trade off γ against ξ: if fewer tuples are marked, then it might be possible to introduce greater changes in the values of marked attributes and hence increase the robustness of a mark to malicious attacks. This trade-off is discussed in greater detail in Sect. 4.

We denote by $r.X$ the value of attribute X in tuple $r \in R$. Figure 1 summarizes the important parameters used in our algorithms. These algorithms make use of cryptographic pseudorandom sequences, which we now briefly review.

---

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

[2] We need not 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.

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
$1/\gamma$	Target fraction of tuples marked
ω	Actual number of tuples marked
α	Significance level of the test for detecting a watermark
τ	Threshold parameter for detecting a watermark

Fig. 1. Notation

// The secret key $\mathcal{K}$ is known only to the owner of the database.
// The parameters γ, ν, and ξ are also private to the owner.
// next($\mathcal{G}$) gives the next number in the random sequence.

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

7)  mark(random_number i, value v, bit_index j) return value

8)      if (i 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
```

Fig. 2. Watermark insertion algorithm

3.1 Cryptographic pseudorandom sequences

A cryptographically secure pseudorandom sequence generator $\mathcal{G}$ deterministically generates a sequence of numbers in which it is computationally infeasible to predict the next number in the sequence [18]. Statistically, the numbers generated by $\mathcal{G}$ appear to be a realized sequence of independent and identically distributed random variables, in the sense that the numbers pass standard statistical tests for these properties [14]. The values in the sequence are determined by the value of an initial seed. Given a fixed seed value, repeated executions of $\mathcal{G}$ generate the same fixed sequence of numbers every time. Several pseudorandom sequence generators are described in [18].

3.2 Watermark insertion

Figure 2 gives the watermark insertion algorithm.[3] At line 2 the random sequence generator $\mathcal{G}$ is initialized with the primary key of the tuple concatenated with the secret key $\mathcal{K}$. Therefore, the sequence of numbers generated for a tuple does not

depend on the sequence generated for any other tuple. This property makes watermarking insensitive to the ordering of tuples. Because a secret key is used to seed $\mathcal{G}$, only someone with knowledge of this key can generate the same sequence of numbers for a given tuple. This sequence is used to decide whether or not the tuple should be marked. If the decision is to mark the tuple, then the sequence is further used to determine the attribute and the bit position within the attribute to be marked as well as the value (0 or 1) of the mark.

Line 3 determines if the tuple under consideration will be marked. For a selected tuple, line 4 determines the attribute that will be marked among the ν candidate attributes. For a selected attribute, line 5 determines the bit position that will be marked among ξ least significant bits. The mark() subroutine sets the selected bit to 0 or 1 depending on the next number in the random sequence. Thus, 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.

Thus, for erasing a watermark precisely, the attacker needs to correctly guess not only the tuples that have been marked but also the marked attribute within a tuple, the bit position that has been marked, and the value of the mark.

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

Suppose that Alice wishes to determine whether or not the relation S published by Mallory has been pirated from her relation R. We allow the sets of tuples and of attributes in S to be strict subsets of those in R. We assume, however, that Mallory does not drop the primary key attribute or change the value of primary keys because the primary key contains valuable information and changing it will render the database less useful.[4]

Watermark detection proceeds as follows. Alice first identifies the bits that should have been marked by the insertion algorithm, basically by executing the operations described in lines 1 through 5 of the insertion algorithm using the original secret key. For each identified bit, Alice tests whether or not the bit's value matches the value that should have been assigned by the insertion algorithm and counts the number of matches. If there are very many matches, then Alice suspects piracy. Alice also suspects piracy if there are suspiciously few matches. For example, if there are no matches at all, then Alice suspects that Mallory has somehow identified the marked bits and systematically flipped each one.

More precisely, Alice models the insertion algorithm as setting bit values according to independent tosses of a fair coin. If the detection algorithm selects ω bits for testing, then the number of matches has a binomial distribution with parameters ω and $1/2$ under the "null" hypothesis that relation S has not been pirated. Thus Alice expects to see roughly $\omega/2$ matches in the absence of piracy. Suppose that she actually observes m matches. Alice rejects the null hypothesis and

---

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

[4] If this assumption does not hold, we use the technique in Sect. 3.6.
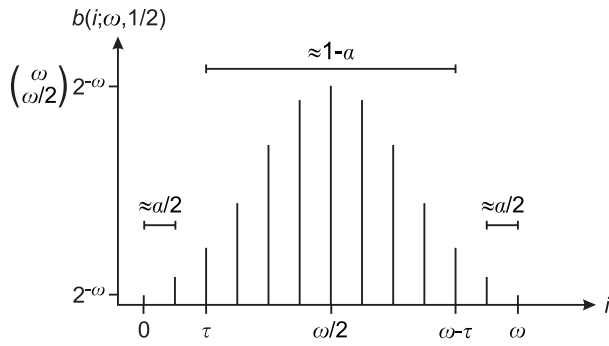
Fig. 3. Definition of τ

// $\mathcal{K}$, 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 $s \in S$ **do**
3)   seed $\mathcal{G}$ with $s.P$ concatenated with $\mathcal{K}$
4)   **if** (next($\mathcal{G}$) mod γ equals 0) **then**   // tuple was marked
5)   attribute_index i = next($\mathcal{G}$) mod ν // A_i was marked
6)   bit_index j = next($\mathcal{G}$) mod ξ // j^{th} bit was marked
7)   totalcount = totalcount + 1
8)   matchcount = matchcount + match(next($\mathcal{G}$), $s.A_i$, j)

9) τ = threshold(totalcount, α)
10) **if** ((matchcount $< \tau$) **or** (matchcount $>$ totalcount $- \tau$)) **then**
11)   *suspect piracy*

12) match(random_number i, value v, bit_index j) **return** integer

13)   **if** (i is even) **then**
14)   return 1 if j^{th} least significant bit of v is 0 else return 0
15)   **else**
16)   return 1 if j^{th} least significant bit of v is 1 else return 0

Fig. 4. Watermark detection algorithm

---

suspects piracy if m is so large or so small that the probability of seeing m matches under the null hypothesis is highly unlikely. Specifically, Alice fixes a small value $\alpha \in (0, 1)$ and sets

$$\tau = \max\left\{ t \in [0, \omega/2]: \sum_{i=t}^{\omega-t} b(i; \omega, 1/2) \geq 1 - \alpha \right\} \qquad (1)$$

where

$$\binom{b(i; n, p) =}{nkp^i(1-p)^{n-i};}$$

(see Fig. 3). She then suspects piracy if either $m < \tau$ or $m > \omega - \tau$, since the probability of so few or so many matches under the null hypothesis is less than or equal to α. The parameter α is called the *significance level* of the test.

The watermark detection algorithm is shown in Fig. 4. Line 3 seeds $\mathcal{G}$ with the primary key of the tuple concatenated with the owner's secret key. This ensures that the sequence of random numbers generated for a tuple will be identical to the one used while marking the relation. Line 4 determines if the

tuple s under consideration should have been marked at the time of inserting the watermark. Lines 5 and 6 determine the attribute and the bit that should have been marked. The subroutine match() then compares the current bit value with the value that should have been set for that bit by the watermarking algorithm.

We thus know at line 9 how many tuples were tested (totalcount) and how many of them contain the expected bit value (matchcount). The subroutine threshold() computes τ as in (1), and the number of matches is compared to the critical values τ and (totalcount $- \tau$) in line 10.

Figure 4 assumes for simplicity that all the candidate attributes $A_0, \ldots, A_{\nu-1}$ are present in S. If Alice finds a tuple s in which she should have marked the attribute A_i (line 5), but Mallory has omitted A_i, she simply ignores the tuple, so that neither matchcount nor totalcount is incremented. 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.

Observe that, by design, the probability of a false hit is less than or equal to α. Therefore, by setting α to a sufficiently small value, Alice can satisfy the accuracy desideratum given in Sect. 2.2.

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 only on its primary key attribute. Thus a tuple can be deleted, inserted, or updated without examining or altering any other tuple. Indeed, a deleted tuple requires no processing by the watermarking algorithm, and an inserted tuple is simply marked as needed. When updating the primary key attribute of a tuple, we recompute the tuple's marking before storing it in the database. When updating a nonprimary key attribute, nothing need be done if the algorithm has not selected this attribute for marking. On the other hand, if the algorithm has selected the attribute, the mark is applied to the attribute's value before the tuple is stored 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 were required for detecting a watermark.

3.5 Extensions

We have assumed up to now that any two attributes are marked at the same rate. This need not be the case. Alice may choose to mark different attributes at different rates because the attributes may tolerate different error rates and, if Alice hides

the rate parameters, Mallory has additional opportunities to make mistakes. Similarly, Alice may vary, from one attribute to another, the number of bits available for marking. Again, the reason could be that different attributes may tolerate different levels of error and Alice may want to create an additional source of mistakes for Mallory. To handle these extensions, the basic algorithm can be modified as follows.

Varying the marking rates   Associate with each attribute A_i a probability p_{A_i} that reflects the relative marking rate for A_i; the probabilities satisfy the usual requirement that $\sum_{i=0}^{\nu-1} p_{A_i} = 1$. Modify line 4 of the watermark insertion algorithm so that next($\mathcal{G}$) is used to probe this probability distribution and select the attribute; for the cost of an extra invocation of the next() operator, the attribute can be quickly selected using the Alias Method [15]. Modify line 5 of the watermark detection algorithm to use the same attribute selection procedure.

Varying the number of candidate bit positions   Replace the global ξ parameter with ν parameters $\xi_{A_1}, \ldots, \xi_{A_\nu}$. Modify lines 5 and 6 of the watermark insertion and detection algorithms, respectively, to use ξ_{A_i} instead of ξ.

3.6 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 a primary key to be present in most of the relations that someone would be interested in watermarking. We now discuss how to extend our technique when a primary key is not available.

Assume first that the relation R consists of a single numeric attribute A, and partition the bits of the attribute A into two groups. Use χ bits of the value $r.A$ in place of the primary key and the remaining ξ for marking in the insertion and detection algorithms. Unlike a primary key, it is possible for the χ bits to have identical values in two different tuples. The presence of duplicate values does not prevent either watermark insertion or detection. However, the χ bits should be chosen such that these duplicates are minimized because an excessive number of duplicates will result in many identical marks that an attacker can potentially exploit.

If the relation has multiple numeric attributes, use one of them in place of the primary key and the remainder for marking. Choose the attribute that has minimum duplicates to serve as the substitute. Alternatively, the concatenation of bits from multiple attributes can be used in place of the primary key in order to reduce duplicates. The drawback to this latter approach is that the watermark cannot be detected if Mallory deletes any one of these attributes.

3.7 Related work

Dugley and Roche [8] 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 secret key and the primary key to select the bit positions. We do the hiding in the original space (i.e., no wavelet transformations, etc.). 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 technique that is closest to ours in the Dugley-Roche taxonomy 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 one unit while correspondingly decreasing the brightness at b_i. The patchwork algorithm is not readily adaptable to database applications, for two reasons. First, completely random changes to relational data can potentially introduce large errors. Second, it is not clear how to handle incremental updates and how to protect the watermark from various forms of attacks.

Another related method 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 of robustness

We now analyze the robustness of our watermarking technique against various forms of malicious attacks and benign updates. Suppose that Alice has fixed a gap value of γ and that the insertion algorithm has marked ω out of η tuples in table R. We model ω as a random variable having a binomial distribution with parameters η and $1/\gamma$ so that the algorithm marks η/γ tuples on average. To detect her watermark, Alice uses a significance level of α. Assume $\alpha = 0.01$ unless stated otherwise. In this and the following two sections, we assume an equal marking rate and the same value of ξ for all attributes.

4.1 Subsetting attacks

Mallory may try to destroy the watermark by subsetting either tuples or attributes. As discussed below, these two attacks are closely related.

4.1.1 Subsetting tuples

The tuple-based subsetting attack rests on the following observation: if exactly ω tuples are selected for testing and the significance level α is such that $\alpha/2 < 2^{-\omega}$, then the definition in Eq. 1 implies that $\tau = 0$, so that the test for detecting the watermark is incapable of yielding a positive result. In

other words, Mallory can successfully destroy the watermark if he can take a subset of Alice's table R that contains less than $\omega_{\min}(\alpha)$ marked tuples, where

$$\omega_{\min}(\alpha) = 1 - \lfloor \log_2 \alpha \rfloor$$

Note that if $\alpha = 10^{-m}$, then $\omega_{\min}(\alpha) \approx 3.3m + 1$.

Suppose that Alice has marked ω tuples and Mallory takes a Bernoulli(κ) sample of the η tuples in R. In other words, Mallory includes each tuple t in the sample with probability κ and excludes t with probability $1 - \kappa$, independent of the other tuples, so that the sample contains $\kappa\eta$ tuples on average. Most current relational database systems support this type of sampling. Given that Alice initially marked ω tuples, the number M of marked tuples in Mallory's sample has a binomial distribution, that is, $P\{M = i \mid \omega\} = b(i; \omega, \kappa)$.[5] It follows that the conditional probability of a successful attack is computed as

$$\begin{aligned} P\{\text{success} \mid \omega\} &= P\{M < \omega_{\min}(\alpha) \mid \omega\} \\ &= B\big(\omega_{\min}(\alpha) - 1; \omega, \kappa\big) \end{aligned}$$

where $B(k; n, p) = \sum_{i=0}^{k} b(i, n, p)$ is the binomial cumulative distribution function.[6] The "-1" in the rightmost term arises from the fact that M takes on only integer values.

As mentioned previously, we model the number ω of marked tuples in R as a binomial random variable with parameters η and $1/\gamma$. It is not hard to see – and can be rigorously shown using, for example, an argument based on moment-generating functions – that taking a Bernoulli($1/\gamma$) sample of the η tuples in the database and then taking a further Bernoulli(κ) subsample is statistically equivalent to directly taking a Bernoulli(κ/γ) sample from the database. It follows that the unconditional distribution of M is binomial with parameters η and κ/γ. The unconditional probability of a successful subsetting attack is therefore

$$P\{\text{success}\} = B\big(\omega_{\min}(\alpha) - 1; \eta, \kappa/\gamma\big)$$

Note that the foregoing analysis also applies to the case in which a fraction κ of tuples is deleted from R in the course of ordinary processing.

Figure 5 displays the probability of a successful subsetting attack for various values of η/γ and κ. Additional experiments (not reported here) indicate that the probability of a successful attack depends on the database size primarily through the ratio η/γ and is relatively insensitive to the absolute size η of the database, so we only show results for the case $\eta = 10^6$. Provided that Alice sets γ so that the watermark insertion algorithm marks about 100 tuples, Mallory's attack will likely fail if he takes more than 10% or so of Alice's tuples. If the insertion algorithm marks roughly 1000 tuples, Mallory's attack will almost certainly fail if he takes more than 2.5% or so of Alice's tuples. The number of tuples that Alice needs to mark is essentially independent of the absolute number of tuples in the database. Thus the larger the database, the easier it is to protect.

[5] If simple random sampling is used, then M has a hypergeometric distribution instead; the qualitative results of our analysis do not change, however.

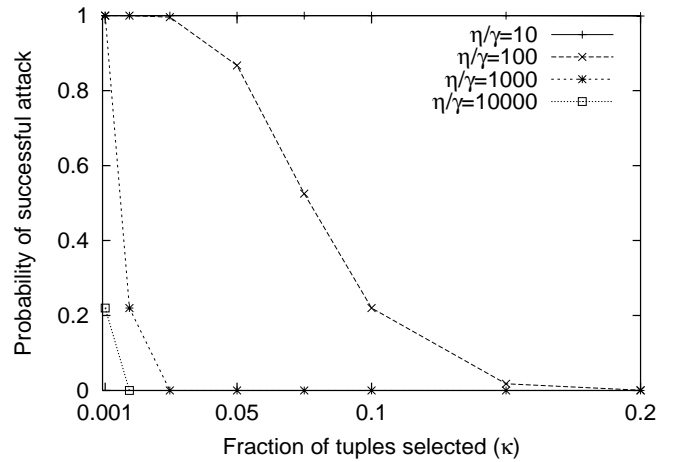[6] We take $B(k; n, p) = 0$ for $k < 0$ and $B(k; n, p) = 1$ for $k > n$.

Fig. 5. Probability of a successful subsetting attack ($\eta = 10^6$, $\alpha = 0.01$)

4.1.2 Subsetting attributes

With respect to the watermark detection algorithm, removing a fraction π of the ν attributes that are candidates for marking is tantamount to removing those marked tuples whose marked attribute belongs to the set of removed attributes. Given that Alice initially marked ω tuples, we can model the number of marked tuples that "remain" in R as a binomial random variable with parameters ω and $1 - \pi$. Arguing as in the case where Mallory subsets tuples, we see that the conditional probability of a successful attack is

$$P\{\text{success} \mid \omega\} = B\big(\omega_{\min}(\alpha) - 1; \omega, 1 - \pi\big)$$

The unconditional probability of a successful attack is

$$P\{\text{success}\} = B\big(\omega_{\min}(\alpha) - 1; \eta, (1 - \pi)/\gamma\big)$$

Thus the analysis is the same as for the tuple-based subsetting attack, but with $1 - \pi$ playing the role of κ.

4.2 Bit-flipping attacks

In this form of attack, Mallory tries to destroy Alice's watermark by altering the value of the bits that he guesses have been marked. Suppose that Mallory magically knows the values of the ν and ξ parameters used by Alice. Since Mallory does not know which bits have been marked, he randomly selects a Bernoulli(κ) sample from the η tuples. For each selected tuple, he flips all of the bits in all ξ bit positions in all ν attributes. It is important to bear in mind that when Alice marks a tuple, she only perturbs a single bit, whereas Mallory perturbs $\xi \times \nu$ bits, thereby inflicting much more damage on each tuple that he marks.

4.2.1 Deterministic bit-flipping

First suppose that Mallory flips each bit deterministically, i.e., that he changes the value x of a bit to $1 - x$. For this type of attack to be successful, he needs to flip between τ and $\omega - \tau$ marked bits. Equivalently, the attack succeeds if and only if Mallory includes between τ and $\omega - \tau$ marked tuples
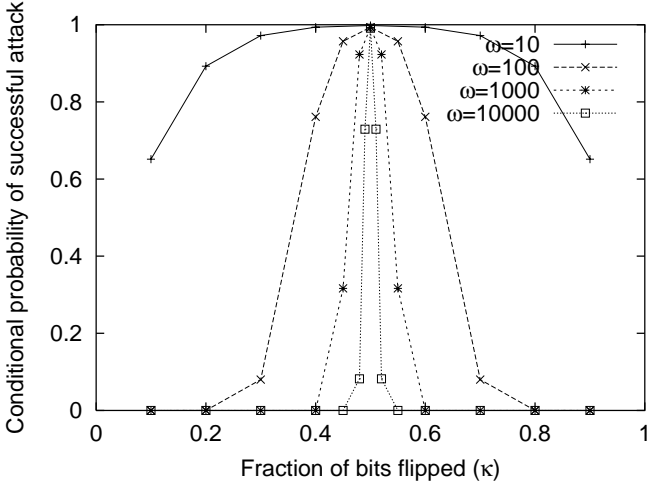
Fig. 6. Conditional probability of a successful deterministic bit-flipping attack ($\alpha = 0.01$)

Fig. 7. Conditional probability of a successful randomized bit-flipping attack ($\alpha = 0.01$)

in the Bernoulli(κ) sample. As in the analysis of the tuple-based subsetting attack, the number M of marked tuples in the sample has a binomial distribution with $P\{M = i \mid \omega\} = b(i; \omega, \kappa)$. The conditional probability of a successful attack, given that Alice initially marked ω tuples, is therefore

$$
\begin{aligned}
P\{\text{success} \mid \omega\} &= P\{\tau \leq M \leq \omega - \tau \mid \omega\} \\
&= \sum_{i=\tau}^{\omega-\tau} b(i; \omega, \kappa) \\
&= B(\omega - \tau; \omega, \kappa) - B(\tau - 1; \omega, \kappa)
\end{aligned}
$$

Observe that the conditional probability of a successful attack does not depend upon the size η of the database.

Figure 6 displays $P\{\text{success} \mid \omega\}$ for various values of ω and κ. The figure clearly shows that Mallory's attack will fail if he attacks either too few or too many tuples. Indeed, we can assume that Mallory will never choose κ larger than 0.5, since for $\kappa > 0.5$ he can achieve the same level of effectiveness while greatly reducing the damage to the data by using a sampling rate of $1 - \kappa$. Overall, Alice's watermark is highly resistant to Mallory's attack: provided that Alice marks at least 1000 tuples, then Mallory's attack will fail unless he damages 40% to 50% of Alice's tuples. As with the subsetting attack, the number of tuples that Alice must mark for a given degree of protection is independent of the size of the database.

4.2.2 Randomized bit flipping

Now suppose that Mallory flips each attacked bit randomly, i.e., sets each bit equal to 0 or 1 according to the independent toss of a fair coin. In effect, Mallory is trying to replace the watermark with random noise. The number M of tuples included in the sample has a binomial distribution as before. The number of matches that Alice observes when she runs the watermark detection algorithm is $K = N + \omega - M$, where N is the number of matches among the M marked tuples in Mallory's sample. Observe that, given M, the random variable N has a binomial distribution with parameters M and $1/2$. It
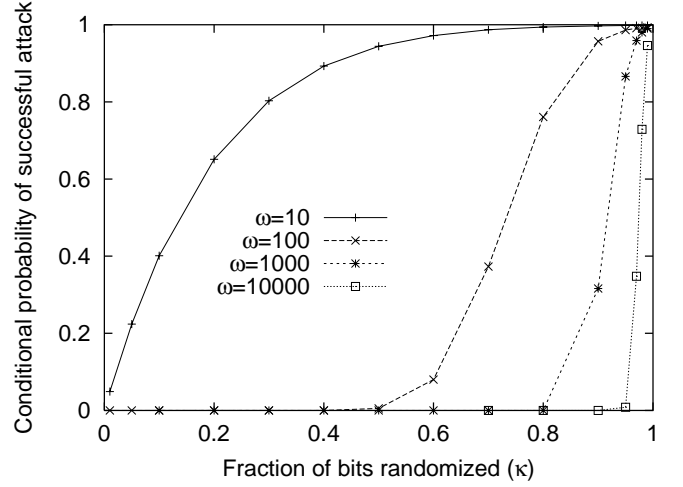
follows that the conditional probability of a successful attack, given that ω tuples have been marked, is

$$
\begin{aligned}
&P\{\text{success} \mid \omega\} \\
&= P\{\tau \leq K \leq \omega - \tau \mid \omega\} \\
&= \sum_{i=\tau}^{\omega} P\{M = i \mid \omega\} \\
&\qquad \times P\{\tau \leq K \leq \omega - \tau \mid \omega, M = i\} \\
&= \sum_{i=\tau}^{\omega} P\{M = i \mid \omega\} \\
&\qquad \times P\{\tau + M - \omega \leq N \leq M - \tau \mid \omega, M = i\} \\
&= \sum_{i=\tau}^{\omega} b(i; \omega, \kappa) \\
&\qquad \times \Big[B(i - \tau; i, 1/2) - B(\tau + i - \omega - 1; i, 1/2)\Big]
\end{aligned}
$$

Similar calculations can be used to compute the probability that the watermark is erased due to random errors in the course of ordinary data processing. However, this probability is much lower than the probability of erasure due to a successful randomized bit-flipping attack as described above.

Figure 7 displays $P\{\text{success} \mid \omega\}$ for various values of ω and κ. If Alice marks at least 100 tuples (regardless of the size of the database), then Mallory's attack will have a nonnegligible probability of success only if he damages at least 50% of the tuples. Because randomizing a marked bit will only erase the mark with 50% probability, the randomized bit-flipping attack has a lower probability of success than the deterministic attack for any fixed values of $\omega > 0$ and $\kappa \in [0, 0.5]$. For example, if Alice marks 100 tuples and Mallory attacks 30% of the tuples in R, then Mallory's probability of success is about 0.08 for a deterministic attack but only about 6×10^{-8} for a randomized attack.

Figure 8 displays results similar to those in Fig. 7, but with $\alpha = 0.0001$. Decreasing α degrades the robustness of the watermarking system in that the probability of a successful attack increases for each value of ω and κ. This effect is relatively mild, however, as long as ω is not too small: even though α
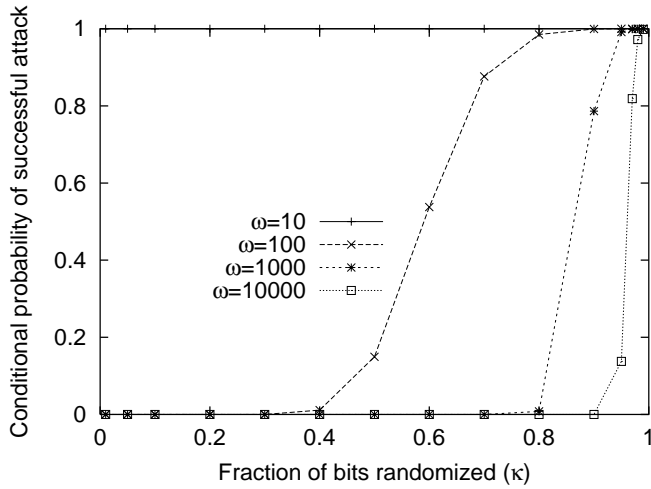
Fig. 8. Conditional probability of a successful randomized bit-flipping attack ($\alpha = 0.0001$)

has decreased by two orders of magnitude, Mallory still needs to damage at least 40% of the data, provided that Alice has marked at least 100 tuples.

In practice, the value of ξ and/or ν will probably be unknown to Mallory. Suppose, for example, that $\nu = 1$ and that Mallory somehow knows the identity of this attribute but needs to guess the value of ξ. If Mallory overestimates ξ, then the results in Figs. 7 and 8 still hold (although the degree of damage to the data increases). Suppose, then, that Alice has marked ω tuples and that Mallory underestimates ξ by m bits, where $0 \le m \le \xi - 1$. For simplicity, we assume that exactly ω_1 tuples are marked in one of bit positions $1, 2, \ldots, \xi - m$ and exactly ω_2 tuples are marked in one of bit positions $\xi - m + 1, \xi - m + 2, \ldots, \xi$, where

$$\omega_1 = \left\lceil \left(\frac{\xi - m}{\xi} \right) \omega \right\rceil \quad \text{and} \quad \omega_2 = \left\lfloor \left(\frac{m}{\xi} \right) \omega \right\rfloor$$

Call the first set of marked tuples "type-1" tuples and the second set "type-2" tuples. Observe that a type-2 tuple is invulnerable to Mallory's attack. For $m \ge 1$, the conditional probability of a successful randomized bit-flipping attack can now be derived using an argument almost identical to the argument for the case $m = 0$. The main difference is that the random variable M now denotes the number of type-1 tuples included in Mallory's sample and the random variable N denotes the number of matches among the M type-1 tuples in Mallory's sample. Thus we have

$$P\{ \text{success} \mid \omega \}$$
$$= \sum_{i=\tau}^{\omega_1} b(i; \omega_1, \kappa)$$
$$\times \left[B(i - \tau; i, 1/2) - B(\tau + i - \omega - 1; i, 1/2) \right]$$

Figure 9 displays $P\{ \text{success} \mid \omega \}$ for various values of m and κ. It can be seen from the figure that underestimating ξ by only a few bits can significantly reduce the probability of a successful attack, even when a large fraction of the tuples are attacked.

In summary, if Mallory overestimates ξ, then he introduces large errors without improving his chances for success; if he underestimates ξ, then his chances of success are reduced.

Note that the larger the value of ξ, the greater the opportunity for Mallory to underestimate ξ. Thus Alice can effectively use the ξ parameter to foil Mallory. (Of course, choosing a ξ that is too large may result in unacceptable data errors.)

Alice has the additional flexibility of marking any one of the ν attributes. The effect on Mallory of having to guess which attributes are marked is similar to the effect of having to guess which bit positions are marked. One key difference is that increasing ν does not increase the perceptibility of the watermark, so Alice should choose ν to be as large as possible.

4.3 Mix-and-match attack

In the mix-and-match attack, Mallory takes a Bernoulli(κ) sample of the η tuples in Alice's relation R to form a subset $R' \subset R$. He then takes a Bernoulli($1 - \kappa$) sample U' from a collection U of η tuples from other sources and mixes the tuples in U' and R' to create his relation S of approximately the same size as R.

As before, the number M of marked tuples in R' has a binomial distribution with $P\{ M = i \mid \omega \} = b(i; \omega, \kappa)$. Suppose, as a simplifying approximation, that the number of tuples in U that the watermark insertion algorithm would mark is exactly the same as the number marked in R, namely, ω. Then the number of tuples N_0 in U' tested by the detection algorithm has a binomial distribution with parameters ω and $1 - \kappa$. Given N_0, the number of matches N among the N_0 tested tuples in U' has a binomial distribution with parameters N_0 and $1/2$. Note that M is conditionally independent of N_0 and N, given ω. In the following, we write $\tau = \tau(\alpha, \omega)$ to make explicit the dependence of the threshold parameter on the significance level and the number of tuples being tested.

To compute the probability of a successful mix-and-match attack, first observe that, for $0 \le i, j \le \omega$ such that $i + j \ge \omega_{\min}(\alpha)$, we have

$$P\{ \text{success} \mid \omega, M = i, N_0 = j \}$$
$$= P\{ \tau(\alpha, i + j) \le M + N \le \omega - \tau(\alpha, i + j) \mid \omega, M = i, N_0 = j \}$$
$$= P\{ \tau(\alpha, i + j) - i \le N \le \omega - \tau(\alpha, i + j) - i \mid \omega, N_0 = j \}$$
$$= B(\omega - \tau(\alpha, i + j) - i; j, 1/2)$$
$$\quad - B(\tau(\alpha, i + j) - i - 1; j, 1/2)$$
$$\stackrel{\text{def}}{=} f(i, j; \omega, \alpha)$$

On the other hand, for i and j such that $i + j < \omega_{\min}(\alpha)$, we have

$$P\{ \text{success} \mid \omega, M = i, N_0 = j \} = 1$$

as in our analysis of the tuple-based subsetting attack. Using the conditional independence of M and N_0, we also have

$$P\{ M = i, N_0 = j \mid \omega \}$$
$$= P\{ M = i \mid \omega \} P\{ N_0 = j \mid \omega \}$$
$$= b(i; \omega, \kappa) b(j; \omega, 1 - \kappa)$$
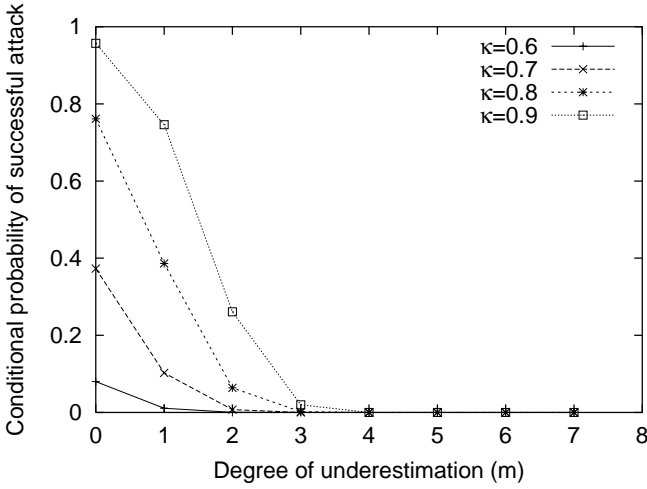$$\stackrel{\text{def}}{=} g(i, j; \kappa, \omega)$$

Fig. 9. Conditional probability of a successful randomized bit-flipping attack when ξ is underestimated ($\alpha = 0.01$, $\omega = 100$, $\xi = 8$)
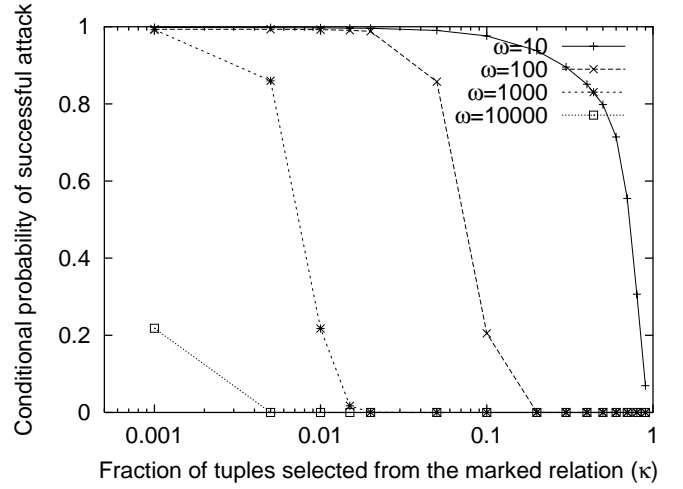

Fig. 10. Conditional probability of a successful mix-and-match attack ($\alpha = 0.01$)

Putting these results together, we find that

$$P\{\text{ success }|\ \omega\}$$
$$= \sum_{i,j} P\{\text{ success }|\ \omega, M = i, N_0 = j\}$$
$$\times P\{M = i, N_0 = j\ |\ \omega\}$$
$$= \sum_{i+j<\omega_{\min}(\alpha)} g(i,j;\kappa,\omega)$$
$$+ \sum_{i+j\geq\omega_{\min}(\alpha)} f(i,j;\omega,\alpha)g(i,j;\kappa,\omega),$$

where, as before, i and j range between 0 and ω. For purposes of computation, note that

$$\sum_{i+j<\omega_{\min}(\alpha)} g(i,j;\kappa,\omega)$$
$$= \sum_{i=1}^{\beta_1} B\big(\omega_{\min}(\alpha) - i - 1; \omega, 1-\kappa\big) b(i;\omega,\kappa)$$

and

$$\sum_{i+j\geq\omega_{\min}(\alpha)} f(i,j;\omega,\alpha)g(i,j;\kappa,\omega)$$
$$= \sum_{i=1}^{\omega} b(i;\omega,\kappa)$$
$$\times \left[\sum_{j=\beta_2(i)}^{\omega} f(i,j;\omega,\alpha)b(j;\omega,1-\kappa)\right]$$

where $\beta_1 = \min\big(\omega, \omega_{\min}(\alpha) - 1\big)$ and $\beta_2(i) = \max\big(\omega_{\min}(\alpha) - i, 0\big)$.

Figure 10 displays $P\{\text{ success }|\ \omega\}$ for various values of ω and κ. If Alice marks 100 (or 1000) tuples, then Mallory's attack will almost certainly fail if he includes more than about 20% (or 3%) of Alice's tuples in his pirated relation.

4.4 False claims of ownership

Mallory may try to claim ownership by simply inserting his watermark in Alice's data; we call this strategy an "additive attack." The competing ownership claims can be resolved if there exist one or more bits that both Alice and Mallory have marked, each with a different value. The idea is simply to determine which owner's marks win. The winner must have overwritten the loser's bits and hence must have inserted the watermark later.

Unfortunately, Mallory can, with little difficulty, ensure that the odds of finding such "conflict" bits are low. Suppose, for example, that Alice has initially marked ω_A bits and then Mallory executes the watermark insertion algorithm with parameters γ_M, ν_M, and ξ_M. Under our usual probabilistic model of the bit-marking process, the probability that a specified bit marked by Alice is also marked by Mallory is the product of the probabilities that the tuple containing the bit is chosen for marking ($= 1/\gamma_M$), that the attribute containing the bit is also chosen ($= 1/\nu_M$), and that the specified bit is chosen ($= 1/\xi_M$). The probability that Mallory's mark is different from Alice's is $1/2$, so that the overall probability that the specified bit is a conflict bit is $(2\gamma_M\nu_M\xi_M)^{-1}$. Because tuples are marked independently of each other, the probability that Mallory "succeeds," i.e., that no conflict bits are found, is

$$P\{\text{ success }|\ \omega_A\} = \left(1 - \frac{1}{2\gamma_M\nu_M\xi_M}\right)^{\omega_A}.$$

For example, if Alice marks 1000 bits and Mallory sets $\gamma_M = 10000$, $\nu_M = 10$, and $\xi_M = 5$, then $P\{\text{ success }|\ \omega_A\} = (1 - 10^{-6})^{1000} \approx 0.999$

As an alternative to adding his own watermark to Alice's database, Mallory might try to claim ownership by finding a key for which the watermark detection algorithm fortuitously detects the presence of a watermark. This type of "invertibility attack" [7] rests on the following observation. Suppose that Mallory fixes values of α, ν, and ξ and repeatedly executes the detection algorithm on Alice's database, using a different key each time. Then, by construction, the algorithm will test positive for a watermark in about $100\alpha\%$ of the runs. Mallory

can choose one of these spurious watermarks and claim that he had originally inserted it in the data.

There are several methods for dealing with the watermarking system's potential vulnerability to false claims of ownership. One method for thwarting an additive attack is to ask both Alice and Mallory to produce, if available, the original copy of the data before the watermark was introduced. Alice can demonstrate the presence of her watermark in Mallory's original database, whereas Mallory cannot show the presence of his watermark in Alice's original database. In general, conflicting ownership claims can be resolved using a secure append-only registry administered by a trusted third party. When Alice initially publishes her data, she appends her key to the registry. If Mallory later claims ownership of the database, the third party resolves the dispute by determining that (1) Alice's watermark is present in the data and (2) Alice's key was appended to the registry before Mallory's key. As a final observation, we note that the benefit to Mallory from successfully establishing a false ownership claim is not as great as the benefit from destroying the watermark by means of a successful malicious attack. Indeed, if the ownership claims cannot be resolved, then customers may be wary of using contested data, thereby reducing the value of the pirated database to Mallory.

5 Analysis of imperceptibility

To gain some intuition about the effects of the various algorithm parameters on the perceptibility of the watermark, suppose that we wish to calculate the mean and variance of the values of an integer-valued attribute. We model the watermark insertion process by assuming that the original attribute values are $x_1, x_2, \ldots, x_\eta$ and the values after insertion of the watermark are $x_1 + \Delta_1, x_2 + \Delta_2, \ldots, x_\eta + \Delta_\eta$, where $\{\Delta_1, \Delta_2, \ldots, \Delta_\eta\}$ are independent and identically distributed (iid) random variables that represent the perturbations caused by watermarking. We further assume that each Δ_i has the representation $\Delta_i = L_i S_i 2^{U_i}$, where, for $1 \le i \le \eta$,

$$P\{L_i = 1\} = 1 - P\{L_i = 0\} = \frac{1}{2\gamma\nu}$$

S_i equals either 1 or -1, each with probability $1/2$, and U_i is uniformly distributed on $\{0, 1, 2, \ldots, \xi - 1\}$. The idea is that $L_i = 1$ if the distinguished attribute for ith tuple is selected for marking, which occurs with probability $(\gamma\nu)^{-1}$, and the new bit value differs from the original bit value, which occurs with probability $1/2$. The definition of the random variable S_i reflects the fact that the attribute value is as likely to be incremented as decremented. Finally, $U_i = k$ if bit k is selected for marking.

The mean attribute value for the original data is $\bar{x} = (1/\eta)\sum_{i=1}^{\eta} x_i$, and the perturbed mean attribute value after insertion of the watermark is $\bar{x} + \bar{\Delta}$, where $\bar{\Delta} = (1/\eta)\sum_{i=1}^{\eta}\Delta_i$. The expected error in computing $\bar{x}$ is

$$\epsilon_M = E[\bar{\Delta}] = 0$$

which follows from the fact that $E[\Delta_i] = 0$ for each i. An easy calculation shows that

$$E[\Delta_i^2] = E\left[L_i 4^{U_i}\right]$$
$$= \frac{1}{2\gamma\nu}\left[\frac{1}{\xi}\left(1 + 4 + 4^2 + \cdots + 4^{\xi-1}\right)\right]$$

$$= \frac{4^\xi - 1}{6\gamma\nu\xi}$$

so that σ_M, the standard deviation of the error, is given by

$$\sigma_M = \text{Var}^{1/2}[\bar{\Delta}] = \frac{E^{1/2}[\Delta_1^2]}{\sqrt{\eta}} \approx \frac{2^\xi}{\sqrt{6\gamma\nu\xi\eta}}$$

With high probability, the error in computing $\bar{x}$ will lie in the range $[-2.5\sigma_E, 2.5\sigma_E]$.

The variance of the original attribute values is given by $V_x = (1/\eta)\sum_{i=1}^{\eta}(x_i - \bar{x})^2$, and after inserting the watermark the perturbed value of the variance is

$$V_{x+\Delta} = \frac{1}{\eta}\sum_{i=1}^{\eta}\left[(x_i + \Delta_i) - (\bar{x} + \bar{\Delta})\right]^2$$

After some algebra, we can write $V_{x+\Delta} - V_x = V_\Delta + 2C_{x,\Delta}$, where

$$V_\Delta = \frac{1}{\eta}\sum_{i=1}^{\eta}(\Delta_i - \bar{\Delta})^2$$

and

$$C_{x,\Delta} = \frac{1}{\eta}\sum_{i=1}^{\eta}(x_i - \bar{x})(\Delta_i - \bar{\Delta})$$

Noting that $E[C_{x,\Delta}] = 0$ and using a standard result for the sample variance, we find that the expected error in computing V_x is given by

$$\epsilon_V = E[V_\Delta] = \left(\frac{\eta - 1}{\eta}\right)E[\Delta_1^2] \approx \frac{2^{2\xi}}{6\gamma\nu\xi}$$

To compute σ_V, the standard deviation of the error, observe that, by (27.4.2) in [6],

$$\text{Var}[V_\Delta] = \frac{E[\Delta_1^4] - E^2[\Delta_1^2]}{\eta} + o\left(\frac{1}{\eta}\right)$$

and it is not hard to show that

$$E[\Delta_1^4] \approx \frac{2^{4\xi}}{30\gamma\nu\xi}$$

$$\text{Var}[C_{x,\Delta}] = \frac{E[\Delta_1^2]V_x}{\eta}$$

and $\text{Cov}[V_\Delta, C_{x,\Delta}] = 0$. It follows that

$$\sigma_V = \text{Var}^{1/2}[V_\Delta + 2C_{x,\Delta}]$$
$$= \left(\text{Var}[V_\Delta] + 4\text{Var}[C_{x,\Delta}]\right)^{1/2}$$
$$\approx \left(\frac{2^{4\xi} + 2^{2\xi+2}V_x}{30\gamma\nu\xi\eta} - \frac{2^{4\xi}}{36\gamma^2\nu^2\xi^2\eta}\right)^{1/2}$$

With high probability, the error in computing V_x will lie in the range $[\epsilon_V - 2.5\sigma_V, \epsilon_V + 2.5\sigma_E]$.

Our analysis implies that the perceptibility of the watermark is highly sensitive to the number of least significant bits selected for marking. It also follows from our formulas that the respective *relative* errors for $\bar{x}$ and V_x become more pronounced when $\bar{x}$ and V_x are small in absolute value. Section 6.2 gives experimental results for a real data set; the results are consistent with our analytical model.

Table 1. Change in variance introduced by watermarking

			γ 10000		1000		100		10	
			ξ 4	8	4	8	4	8	4	8
Attribute	Mean	Variance								
Elevation	2959	78391						+3		+16
Aspect	155	12525						+1		+8
Slope	14	56						+1		+14
Horz-Dist-Hydro.	269	45177						+1	+1	
Vert-Dist-Hydro.	46	3398				+1		+2		+11
Horz-Dist-Roads	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-FPts	1980	1753490				+1		-3		-1

Our formulas also show the trade-off between the parameters γ and ξ: if γ is increased, then ξ can also be increased without increasing the perceptibility of the watermark. As discussed in Sect. 4.2, increasing ξ can improve the robustness of the watermark. Of course, γ must not be increased so far that the number of marked tuples is set dangerously low, or the overall effect on robustness will be negative (see the curves for $\omega = 10$ in Figs. 6–8 and Fig. 10).

6 Experimental results

We now report some experimental results that complement the analysis presented in Sect. 4. Experiments were performed using the Forest Cover Type data set, available from the University of California at Irvine KDD Archive.[7] The data set has 581,012 rows, each with 61 attributes. We added an extra attribute called *id* to serve as the primary key and chose the first ten integer-valued attributes as candidates for watermarking.

We ran experiments on DB2 UDB v. 7 using JDBC connectivity on a Windows NT Workstation 4.0 with a 400-MHz Intel processor, 128 MB of memory, and a 10-GB disk drive. The log file size was set to 20 MB and the lock list to 2 MB.

6.1 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.

The first experiment evaluated the cost of inserting a watermark. We focused on the most expensive scenario by setting γ equal to 1. In this case, the watermarking algorithm reads η tuples and finds that every tuple requires marking. However, on average, half the tuples already have the correct value for the mark. Therefore, the insertion algorithm updates, on average, only $\eta/2$ tuples. We compared the average time (over the 30 replications) to insert a watermark to the average time required to read η tuples and update $\eta/2$ tuples. The comparison yielded a ratio of 1.16. This rather small overhead of 16% is due to the cost of generating the pseudorandom numbers

used to insert the watermark. The total elapsed time required to watermark the relation was 2245 s on average. This time included the cost of logging updates to half of the tuples in the relation.

The second experiment evaluated the cost of detection. We again considered the most expensive case by setting γ equal to 1 and 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. The major component of the cost was again the generation of pseudorandom numbers. The total detection time was 214 s on average.

These results indicate that our algorithms perform well enough to be used in real-world applications.

6.2 Effect on mean and variance

We next evaluated experimentally the impact of watermarking on the mean and variance of values of various marked attributes. Experiments were performed for $\gamma = 10, 100, 1000, 10000$ and $\xi = 1, 4, 8$. We found that, for each attribute, the change in the mean was minuscule in all cases. Table 1 shows the change in the variance for the different attributes. The values are rounded to the nearest integer. An empty entry indicates very little or no change; all entries for $\xi = 1$ are empty and therefore not displayed. As expected, the largest changes in the 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 relative to the original values of the variances. The only significant change occurs in the *Slope* attribute when $\xi = 8$ and $\gamma = 10$. Compared to the other attributes, *Slope* has relatively small values that are perturbed significantly when ξ is large. These results are consistent with the analysis in Sect. 5. Note that if any of the foregoing changes in the variance are deemed unacceptable, then the ν, ξ, and γ parameters can be adjusted to reduce the impact of watermarking on the data.

---

[7] kdd.ics.uci.edu/databases/covertype/covertype.html.

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

Fig. 11. Design trade-offs

7 Summary

The following are the major contributions of this paper:

- Identification of rights management of relational data through watermarking as an important and technically challenging problem for database research.
- Delineation of the desirable properties of a watermarking system for relational data.
- Enumeration of malicious attacks from which the watermark inserted in a relation must be protected.
- Provision of the first watermarking technique specifically geared for relational data.
- Analysis and empirical evaluation of the robustness and effectiveness of the proposed technique to demonstrate the feasibility of watermarking real-life data sets.

Our watermarking technique has four important tunable parameters: (i) α, the test significance level, (ii) γ, the gap parameter that determines the target 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. Figure 11 summarizes the important trade-offs when selecting the values for these parameters.

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

References

1. Atallah M, Wagstaff S (1999) Watermarking with quadratic residues. In: Proceedings of IS&T/SPIE conference on security and watermarking of multimedia contents, January 1999
2. Bender W, Gruhl D, Morimoto N (1995) Techniques for data hiding. In: Proceedings of the SPIE 2420 (storage and retrieval for image and video databases III), pp 164–173
3. Boney L, Tewfik AH, Hamdy KN (1996) Digital watermarks for audio signals. In: Proceedings of the international conference on multimedia computing and systems, Hiroshima, June 1996
4. Collberg CS, Thomborson C (2000) Watermarking, tamper-proofing, and obfuscation – tools for software protection. Technical report 2000–03, University of Arizona
5. Cox IJ, Miller ML (1997) A review of watermarking and the importance of perceptual modeling. In: Proceedings of electronic imaging, February 1997
6. Cramér H (1946) Mathematical methods of statistics. Princeton University Press, Princeton, NJ
7. Craver S, Memon N, Yeo BL, Yeung MM (1998) Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. IEEE J Selected Areas Commun 16(4):573–586
8. Dugelay JL, Roche S (2000) A survey of current watermarking techniques. In: Katzenbeisser S, Petitcolas FA (eds) Information hiding techniques for steganography and digital watermarking. Artech House, Norwood, MA, pp 121–148
9. Hartung F, Girod B (1998) Watermarking of uncompressed and compressed video. Signal Process 66(3):283–301
10. Johnson NF, Duric Z, Jajodia S (2000) Information hiding: steganography and watermarking – attacks and countermeasures. Kluwer, Amsterdam
11. Ó Ruanaidh JJK, Dowling WJ, Boland FM (1996) Watermarking digital images for copyright protection. IEEE Proc Vision Signal Image Process 143(4):250–256
12. Katzenbeisser S, Petitcolas FA (eds) Information hiding techniques for steganography and digital watermarking. Artech House, Norwood, MA
13. Kerckhoffs A (1883) La cryptographie militaire. J Sci Militaires 9:5–38
14. Knuth D (1981) Seminumerical algorithms. In: The art of computer programming, vol 2. Addison-Wesley, Reading, MA
15. Law A, Kelton W (2000) Simulation modeling and analysis, 3rd edn. McGraw-Hill, New York
16. Maes M (1998) Twin peaks: the histogram attack on fixed depth image watermarks. In: Proceedings of the 2nd international workshop on information hiding, Lecture notes in computer science, vol 1525. Springer, Berlin Heidelberg New York, pp 290–305
17. Maxemchuk N (1994) Electronic document distribution. Technical journal, AT&T Labs
18. Schneier B (1996) Applied cryptography, 2nd edn. Wiley, New York
19. Wagner NR (1983) Fingerprinting. In: Proceedings of the IEEE symposium on security and privacy, Oakland, CA, April 1983, pp 18–22