SYSTEM AND METHOD FOR DIGITAL WATERMARKING OF DATA REPOSITORY

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to data repository security.

Description of the Related Art

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. 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 and build parametric databases. 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.

In the context of databases, watermarking poses challenges that are not necessarily present in techniques for watermarking multimedia data, most of which were initially developed for still images and later extended

to video and audio sources. The differences between the two applications, as understood herein, include the following:

1. 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, on the other hand, consists of tuples, each of which represents a separate object. The watermark must be spread over these separate objects.

2. 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.

3. 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. Likewise, watermarking techniques for text, which exploit the special properties of formatted text, cannot be easily applied to databases. Furthermore, techniques for watermarking software have had limited success, because the instructions in a computer program can often be rearranged without altering the semantics of the program. This resequencing can, however, destroy a watermark.

SUMMARY OF THE INVENTION

Accordingly, the invention provides a computer for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer including at least one processor executing method acts comprising: establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

The present invention can preferably be used to identify pirated copies of data.

Database relations which can be watermarked preferably have attributes which are such that changes in some of their values do not affect the applications. Real world datasets exist that can tolerate a small amount of error without degrading their usability. For example, the ACARS meteorological data, which is used in building weather prediction models, has wind vector and temperature accuracies estimated to be within 1.8 m/s and 0.5 C respectively. The present invention in accordance with a preferred embodiment recognizes that errors introduced by watermarking can easily be constrained to lie within the measurement tolerance in such data. As another example, consider experimentally obtained gene expression datasets that are being analyzed using various data mining techniques. Again, the present invention in accordance with a preferred embodiment recognizes that the nature of the data collection and the analysis techniques is such that changes in a few data values is unlikely to 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 few transactions. Finally, consider the parametric data on semiconductor parts mentioned above. For many parameters, errors introduced by watermarking can be made to be within the measurement tolerance.

The data repository may for example be a file system, database, or other record store. The computer preferably establishes watermark values for a bit position in some of the attributes of some of the tuples based on private information, to establish a bit pattern which defines a watermark.

In a preferred embodiment only bit positions in numeric attributes need be part of the watermark. The watermark values preferably are established at least in part based on a one-way hash function of at least the private information.

According to a preferred embodiment the computer can determine whether a test database that is suspected as having been copied from a watermarked database in fact contains the watermark. Using the private information, the computer preferably determines whether the watermark exists in the test data structure within a predetermined probability.

Preferably there is provided a general purpose computer for determining whether a watermark exists in a test data repository having tuples. Private information is preferably used to determine whether the watermark exists in the test data structure.

In one embodiment at least one of: tuples to be marked, attributes to be marked within a tuple, bit positions to be marked in an attribute, and specific bit watermark values, are determined based on the private information.

In one embodiment at least two of: tuples to be marked, attributes to be marked within a tuple, bit positions to be marked in an attribute, and specific bit watermark values, are determined based on the private information.

In one embodiment, all of: tuples to be marked, attributes to be marked within a tuple, bit positions to be marked in an attribute, and specific bit watermark values, are determined based on the private information.

Preferably only bit positions in numeric attributes are part of the watermark.

Preferably the watermark values are established at least in part based on a one-way hash function of at least the private information.

Preferably it is possible to receive a test data structure; and at least in part using the private information, determine whether the watermark exists in the test data structure.

Preferably it is determined whether the watermark exists in the test data structure within a predetermined probability.

In one embodiment, at least one of, at least two of, or all of: tuples to be marked, attribute to be marked within a tuple, bit position to be marked in an attribute, and specific bit watermark value, are determined based on the private information.

According to one aspect, the invention provides a computer for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer including at least one processor executing method acts comprising: receiving a test data structure; and at least in part using private information, determining whether the watermark exists in the test data structure.

Preferably it is determined whether the watermark exists in the test data structure within a predetermined probability.

Preferably, for a data repository from which the test data structure was derived, watermark values are established for at least some bit positions for some of the attributes of some of the tuples based on the private information.

According to a preferred embodiment, there is provided a computer program device comprising: a computer program storage device readable by a digital processing apparatus; and a program on the program storage device and including instructions executable by the digital processing apparatus for promoting data repository security, the program comprising: means for determining plural tuples to be marked based on private information; means for determining at least one attribute to be marked in each tuple to be marked; means for determining at least one bit to be marked in each attribute to be marked; and means for establishing a watermark value for at least each non-null bit to be marked to establish the watermark.

According to a preferred embodiment, there is provided a method for enhancing security of a data repository, comprising: using secret information to establish a pattern of bit values throughout the data repository.

Preferably plural tuples in the data repository to be marked are determined based on private information.

Preferably at least one attribute to be marked in each tuple to be marked is determined.

Preferably at least one bit to be marked in each attribute to be marked is determined.

Preferably there is established a watermark value for at least each non-null bit to be marked to establish the watermark.

According to a preferred embodiment, there is provided a computer program device comprising: a computer program storage device readable by a digital processing apparatus; and a program on the program storage device and including instructions executable by the digital processing apparatus, the program comprising: means for receiving a test data structure having tuples; means for determining, for each tuple, whether the tuple should

have been marked based on private information; means for determining, for at least one attribute in each tuple that should have been marked, whether the attribute should have been marked; means for determining, for at least one bit in each attribute that should have been marked, whether the bit should have been marked; and means for determining a watermark value for each bit that should have been marked to determine whether the test data structure contains a watermark.

Preferably only bit positions in numeric attributes are tested to be part of the watermark.

Preferably it is possible to maintain a count of correct bit values; and based on the count, to determine whether the watermark exists in the test data structure.

Preferably the watermark values are determined at least in part based on a one-way hash function of at least the private information.

According to a preferred embodiment, there is provided a method comprising: using secret information to determine whether a pattern of values exists in a test data structure.

Preferably it is determined, for each tuple in the test data structure, whether the tuple should have been marked based on private information.

Preferably it is determined for at least one attribute in each tuple that should have been marked, whether the attribute should have been marked.

Preferably it is determined for at least one bit in each attribute that should have been marked, whether the bit should have been marked.

Preferably a watermark value is determined for each bit that should have been marked to determine whether the test data structure contains a watermark.

In one embodiment, at least some attributes have primary keys, and the primary key of an attribute is used to establish a watermark value.

In one embodiment, at least first attributes do not have primary keys, and a partition of bits in a first attribute is used to establish a watermark value.

In one embodiment at least some attributes have primary keys, and the primary key of an attribute is used to establish a bit value.

According to one aspect, the invention provides a method for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the method comprising the step of: establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

According to another aspect, there is provided a computer program for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer program comprising program code means, which when executed on a computer, performs the steps of: receiving a test data structure; and at least in part using private information, determining whether the watermark exists in the test data structure.

According to another aspect, there is provided a method for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the method comprising the steps of: receiving a test data structure; and at least in part using private information, determining whether the watermark exists in the test data structure.

According to another aspect, there is provided a computer program for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer program comprising program code means, which when executed on a computer perform the steps of: establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

According to another aspect, there is provided a data repository, comprising: plural tuples, each tuple having at least one attribute, each attribute having at least one bit having a bit value; a value pattern among bits being established based on secret information.

BRIEF DESCRIPTION OF THE DRAWINGS

A preferred embodiment of the present invention will now be described, by way of example only, and with reference to the following drawings:

Figure 1 is a schematic diagram showing a system in accordance with a preferred embodiment of the present invention;

Figure 2 is a flow chart of the logic for establishing a watermark in a database in accordance with a preferred embodiment of the of the present invention; and

Figure 3 is a flow chart of the logic for determining whether a watermark exists in a test data structure in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring initially to Figure 1, a system is shown, generally designated 10. As shown, the system 10 includes a computer 12 that accesses a data structure or repository such as a file system or, in the exemplary embodiment shown in Figure 1, a relational database 14. The computer 12 thus can host a relational database management system. A watermark application 16 can be executed on the computer 12 to undertake the logic shown in Figures 2 and 3.

In an exemplary non-limiting embodiment the watermark application 16 can be a user-defined function written in Java™. The computer 12 may be a Windows NT® Version 4.00 workstation executing DB2® Universal Database™ (UDB) Version 7 using Java™ Database Connectivity (JDBC™ connectivity). Other types of computers, including without limitation main frame computers, laptops, desktops, and notebook computers, may be used, as well as other types of databases. (Java, all Java-based logos and JDBC are trademarks of Sun Microsystems Inc in the United States, other countries or both; Windows NT is a registered trademark of Microsoft Corporation in the United States, other countries, or both; and DB2 and DB2 Universal

Database are trademarks of International Business Machines Corporation in the United States, other countries, or both)

For illustration purposes, Figure 1 shows that the data repository (e.g., the database 14) contains tuples 18, with each tuple 18 containing one or more attributes 20. In turn, each attribute 20 may include one or more data bits 22, and each bit 22 has a value 24 that, e.g., is either "0" or "1" (or, in some cases, null). In the preferred embodiment a pattern of values is established, such as a pattern of bit values or less preferably a pattern of database attribute values, using secret information that consequently cannot be detected except by accessing the secret information.

With the above overview of the present architecture in mind, it is to be understood that the present logic is executed on the architecture shown in Figure 1 in accordance with the flow charts discussed below. The flow charts herein illustrate the structure of the logic of the preferred embodiment as embodied in computer program software. Those skilled in the art will appreciate that the flow charts illustrate the structures of logic elements, such as computer program code elements or electronic logic circuits, that function according to the preferred embodiment. Preferably a machine component renders the logic elements in a form that instructs a digital processing apparatus (that is, a computer) to perform a sequence of function steps corresponding to those shown.

In other words, the logic may be embodied by a computer program that is executed by a processor as a series of computer-executable instructions. These instructions may reside, for example, in RAM or on a hard drive or optical drive, or the instructions may be stored on magnetic tape, electronic read-only memory, or other appropriate data storage device. In an illustrative embodiment of the invention, the computer-executable instructions may be lines of Java or compiled C++ compatible code.

Now referring to Figure 2, the logic for establishing a watermark in the database 14 can be seen. Commencing at block 26, a do loop is entered for each tuple 18 in the database 14 in which loop private information is to be used. While the preferred private information can be a "key" such as the private key of a public key-private key pair, any suitable private information, such as a string of confidential numbers, may be used.

To better understand Figure 2, the following definitions are first provided.

Assume that the database 14 can be regarded as a database relation R whose scheme is $R(P, A_0,\ldots,A_{v-1})$, where P is the primary key attribute. For illustration, assume that all v attributes $A_0,\ldots,A_{v-1}$ are candidates for marking. That is, the database 14 has η tuples and v is the number of attributes in the relation that are available for marking, with ξ being the number of least significant bits available for marking in an attribute. Also, $1/\gamma$ represents roughly the fraction of tuples that will be marked, ϖ represents the number of tuples marked after the logic of Figure 2, α represents the significance level of the test for determining whether a watermark exists in a test data structure as discussed further below in reference to Figure 3, and τ is the minimum number of correctly marked tuples needed for indicating the presence of a watermark as disclosed further below in Figure 3. Preferably, all candidate attributes for marking are numeric attributes and their values are such that changes in ξ least significant bits for all of them are imperceptible. The symbol $r.A_i$ refers to the value of attribute A_i in tuple $r \in R$, the set of available attributes.

In addition to the above, the following explanation of certain functions is provided. The presently preferred non-limiting implementation uses a one-way hash function H to operates on an input message M of arbitrary length, and H returns a fixed length hash value h, i.e., h = H(M). The preferred hash function 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 H(M) = h, and iii) given M, it is hard to find another message M' such that H(M) = H(M'). Message Digest 5 and SHA are two good choices for H.

A message authenticated code (MAC) is a one way hash function that depends on a key. Assume F is 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. F can be seeded with a private key K that is known only to the owner of the database 14. In the preferred embodiment, the following secure MAC can be used:

$F(r.P) = H(K \bigcirc H(K \bigcirc r.P))$, wherein $\bigcirc$ indicates concatenation.

With the above definitions in mind, the description of Figure 2 may be completed. The logic moves to decision diamond 28, wherein in one preferred non- limiting embodiment it is determined whether the tuple

under test should be marked as follows. The tuple is designated for marking if F(r.P) mod γ equals 0. It will readily be appreciated that only the user with a knowledge of the secret information K can easily know what tuples to mark and, hence, what tuples have been marked.

If the test for marking is not met at decision diamond 28, the method loops back to get the next tuple. Otherwise, if the tuple is to be marked, the logic proceeds to block 30 to determine which attribute(s) of the tuple to mark. To do this, in one preferred non- limiting implementation the i^{th} attribute is selected for marking, where attribute_index i = F(r.P) mod v. Then, moving to block 32, for a selected attribute it is determined which bit(s) to mark. To do this, in one preferred non-limiting implementation the j^{th} bit of the attribute is selected for marking, where bit_index j = F(r.P) mod ξ. It is not necessary to use consecutive ξ least significant bits for marking. For instance, those bit positions in which the distribution of bit values is skewed can be omitted if desired.

A hash function is then used at block 34 to determine the value of the selected bit. This value can be referred to as a "watermark value". In one preferred non-limiting implementation, the value of the j^{th} least significant bit is set to "0" if H(K ○ pk) is even, otherwise the bit value is "1", wherein "pk" is the attribute's primary key.

The pseudocode below is an example of the above logic. It is to be understood that the pseudocode is written in a form that simplifies disclosure, rather than in the most computationally efficient form.

```
// The private key K is known only to the owner of the database.
// The parameters γ, v, and ξ are also private to the owner.
1) for each tuple r ∈ R do
2)     if (F(r.P) mod ã equals 0) then // mark this tuple
3)           attribute_index i = F(r.P) mod v // 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
```

With the above in mind, it may now be appreciated that 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, the mark preferably is not applied to the null value, leaving it unchanged. In any case, in one non-limiting implementation, at block 36 the watermark value can be inserted in the bit using an SQL update function. The logic then fetches the next tuple and loops back to decision diamond 28.

As provided for in one non-limiting embodiment, whether a tuple is marked or not depends on its primary key attribute. Consequently, 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, its marking can be recomputed before storing the tuple in the database. When updating a nonprimary 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 preferably can be applied to the attribute value before storing it in the database.

It may be the case that no primary key exists in the relation being watermarked. If this is the case, and assuming that the relation R consists of a single numeric attribute A, the bits of the attribute A can be partitioned into two groups. "X" 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. If the relation has more than one attribute, one of them can be used as the substitute and the remainder for marking. The attribute that has minimum duplicates is selected to serve as the primary key substitute. The substitute can also be spread across more than one attribute to reduce duplicates.

Now referring to Figure 3, the logic by which it can be determined whether a test data structure such as all or a portion of a database might have been copied can be seen. The logic is probabilistic, as will become evident.

Commencing at block 38, a do loop is entered for each tuple in the test data structure, with the private information K being available. At decision diamond 40, it is determined whether the tuple under test was to have been marked as part of an authorized watermarking. To do this, in one preferred, non-limiting implementation it is determined whether F(r.P)

mod γ equals 0. If so, the tuple should have been marked, had it been processed through the algorithm of Figure 2. If the test for marking is not met at decision diamond 38, the method loops back to get the next tuple.

Otherwise, if the tuple should have been marked, the logic proceeds to block 42 to determine which attribute(s) of the tuple would have been marked, had the watermarking of Figure 2 been undertaken. To do this, in one preferred non-limiting implementation the i^{th} attribute is designated as having been selected for marking, where attribute_index i = F(r.P) mod v. Also, the j^{th} bit is designated as having been selected for marking, where bit_index j = F(r.P) mod ξ.

Proceeding to decision diamond 44, the actual value of the bit selected at block 42 is compared to what the value should have been had the data structure been watermarked with the private information K. A count is incremented if there is a match at block 46. At decision diamond 48 it is determined whether the last tuple has been tested. If not, the logic loops back to decision diamond 40, retrieving the next tuple. Otherwise, the logic proceeds to decision diamond 50 to determine whether the count exceeds a threshold Õ. If not, "watermark not found" is returned at block 52. Otherwise, "watermark found (suspect piracy)" is returned at block 54.

The pseudocode below shows one non-limiting implementation of the watermark detection logic.

```
// K, γ, v, 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 the test data structure S do
3)     if (F(s.P) mod γ equals 0) then // this tuple was marked
4)           attribute_index i = F(s.P) mod v // 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) //
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
```

The detection logic of Figure 3 can be performed using a user-defined aggregate function in accordance with the logic shown in Figure 3, with the detection function returning a "1" if a watermark is found and otherwise returning a "0".

In the above pseudocode, the threshold subroutine to be used in line 8 can be specified as follows. Assume the logic of Figure 3 tests ω tuples from the "suspicious" (test) database. These tests can be regarded as Bernoulli trials with a probability of 1/2 that the logic will successfully find the looked-for value in a specific bit position. The subroutine in line 8 returns a minimum τ such that the probability that at least τ successes in ω trials return correct results is less than a. The significance of a determines how amenable the system is to false hits, with the lower the a, the lower the chance of incorrectly identifying an innocent test database as having the watermark.

In the logic of Figure 3, if a tuple attribute which was marked has been omitted, the tuple is ignored. Similarly, if a tuple is found whose attribute should have been marked, but the attribute has a null value, the tuple is ignored. In any case, the values of matchcount and totalcount are unaffected.

It is to be appreciated that the detection algorithm is blind, because it simply extracts bits of information from the data, without requiring access to the original data or watermark to arrive at its decision. This is important for database relations since relations are frequently updated. Without blind detection, each version of the relation would need to be kept because the original would be required for detecting a watermark.

In a particular non-limiting implementation, as mentioned above a watermark can be inserted by first retrieving tuples of R, with attributes P, A being specified in the select list. The select statement can contain the additional clause "for update of A" that allows 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 "r" does not fall into a gap and a change is needed in the value of r.A, an update statement can be issued to mark r.A. The update statement can have a "current of cursor" clause that allows the database engine to know that the tuple to be updated is r.

As also disclosed above, watermark detection can be performed using a select statement to fetch the tuples of the suspicious database relation S, specifying the attributes P, A that are in the select list. If a result tuple "s" does not fall into a gap, the count "totalcount" is incremented. If "s" contains the mark, the count "matchcount" is also incremented. When all the tuples have been processed, the detection algorithm determines whether the probability of finding "matchcount" marks in "totalcount" tuples is within the significance level. If so, the watermark has been detected.

Depending on how tolerant the data is to errors, a value for what can be termed the "gap" γ is selected to force an unscrupulous copyist to commit large errors in attempting to erase the watermark, making the copyist's data less desirable. Gap γ is a control parameter that determines the number of tuples marked, with $\omega \approx \eta/\gamma$. A tradeoff can be made, γ against ξ, that determines the extent of error introduced in an attribute's values. That is, if less tuples are marked, the greater the extent of change that can be made to the values of marked attributes. Also, ξ is selected such that the unscrupulous copyist guesses the value too high, large errors result, whereas if he underestimates the value, the chances of his success in destroying the watermark decrease. v is likewise established to foil attacks by unscrupulous copyists.

In summary, decreasing values of α decrease the chance of false hits but increase the chance that a watermark in a copied, possibly altered database will be missed. Decreased values of γ increase the robustness of the system to attack but increase data errors in the watermarked data. Increased values of v and ξ increase robustness, but in the case of ξ, increases the value of errors in the data.

CLAIMS

1. A computer for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer including at least one processor executing method acts comprising:

establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

2. The computer of Claim 1, wherein at least one of, at least two of, or all of: tuples to be marked, attributes to be marked within a tuple, bit positions to be marked in an attribute, and specific bit watermark values, are determined based on the private information.

3. The computer of Claim 1, wherein only bit positions in numeric attributes are part of the watermark.

4. The computer of Claim 1, wherein the watermark values are established at least in part based on a one-way hash function of at least the private information.

5. A computer for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer including at least one processor executing method acts comprising:

receiving a test data structure; and

at least in part using private information, determining whether the watermark exists in the test data structure.

6. The computer of Claim 5, wherein the determining act is undertaken by determining whether the watermark exists in the test data structure within a predetermined probability.

7. A computer program for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit

value, the computer program comprising program code means, which when executed on a computer perform the steps of:

establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

8. The computer program of claim 7, wherein the step of establishing watermark values comprises the steps of:

determining plural tuples to be marked based on private information;

determining at least one attribute to be marked in each tuple to be marked;

determining at least one bit to be marked in each attribute to be marked; and

establishing a watermark value for at least each non-null bit to be marked to establish the watermark.

9. A method for watermarking a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the method comprising the step of:

establishing watermark values for at least one bit position for at least one attribute of at least some of the tuples based on private information, to establish a watermark.

10. The method of Claim 9, wherein the step of establishing watermark values further comprises:

determining plural tuples in the data repository to be marked based on private information;

determining at least one attribute to be marked in each tuple to be marked;

determining at least one bit to be marked in each attribute to be marked; and

establishing a watermark value for at least each non-null bit to be marked to establish the watermark.

11. A computer program for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the computer program comprising program code means, which when executed on a computer, performs the steps of:

receiving a test data structure; and

at least in part using private information, determining whether the watermark exists in the test data structure.

12. The computer program of claim 11, wherein the receiving step, receives a test data structure having tuples and the step of determining comprises;

determining, for each tuple, whether the tuple should have been marked based on private information;

determining, for at least one attribute in each tuple that should have been marked, whether the attribute should have been marked;

determining, for at least one bit in each attribute that should have been marked, whether the bit should have been marked; and

determining a watermark value for each bit that should have been marked to determine whether the test data structure contains a watermark.

13. The computer program of Claim 12, further comprising program code means, which when executed on a computer, perform the steps of:

maintaining a count of correct bit values; and

based on the count, determining whether the watermark exists in the test data structure.

14. A method for determining whether a watermark exists in a data repository having tuples, each tuple being associated with one or more attributes, each attribute having one or more bit positions, at least some bit positions having a bit value, the method comprising the steps of:

receiving a test data structure; and

at least in part using private information, determining whether the watermark exists in the test data structure.

15. The method of Claim 14, wherein the step of determining comprises:

determining, for each tuple in the test data structure, whether the tuple should have been marked based on private information;

determining, for at least one attribute in each tuple that should have been marked, whether the attribute should have been marked;

determining, for at least one bit in each attribute that should have been marked, whether the bit should have been marked; and

determining a watermark value for each bit that should have been marked to determine whether the test data structure contains a watermark.

16. A data repository, comprising:

plural tuples, each tuple having at least one attribute, each attribute having at least one bit having a bit value;

a value pattern among bits being established based on secret information.

17. The data repository of Claim 16, wherein at least some attributes have primary keys, and the primary key of an attribute is used to establish a watermark value.

18. The data repository of Claim 16, wherein at least first attributes do not have primary keys, and a partition of bits in a first attribute is used to establish a watermark value.