Study on Database Watermarking

Bofin Babu | 2013H313085H

Database Watermarking

- A method for ensuring data integrity.
- Useful for tamper detection.
- Combines aspects of data hashing and digital watermarking

“Watermarking Relational Databases”

Rakesh Agrawal & Jerry Kiernan
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

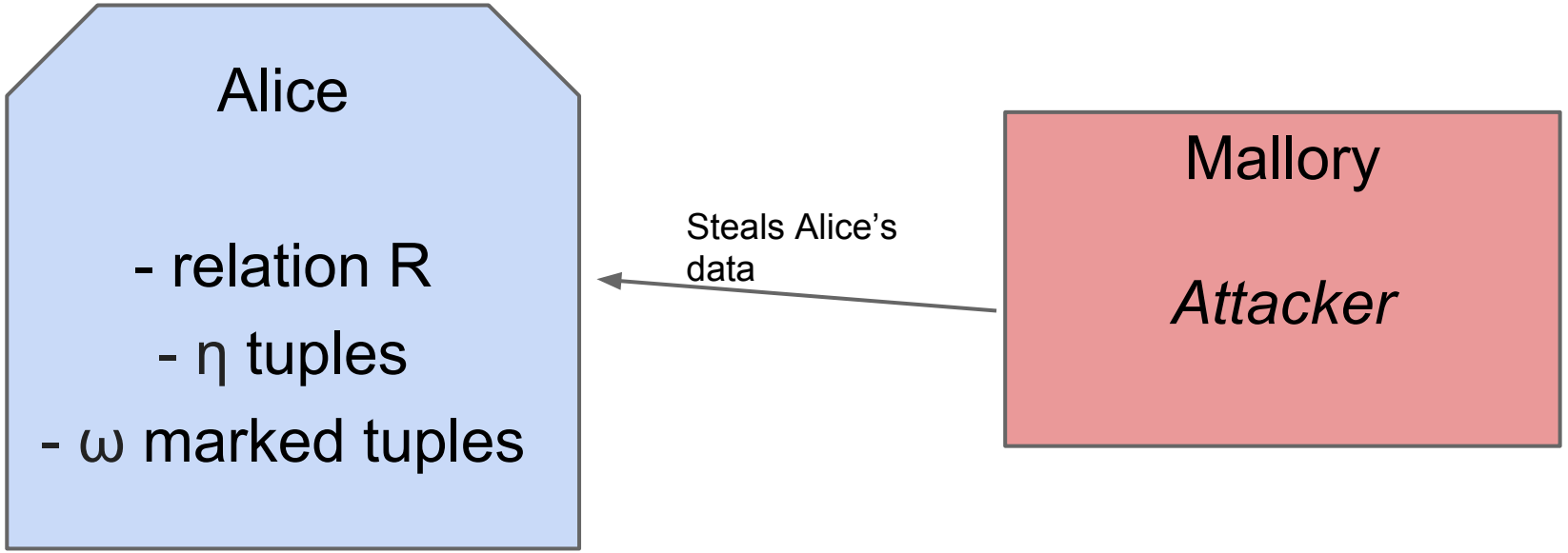
Year: 2002

Citations: around 460

Introduction

- Need for watermarking of databases.
- Real-world datasets that can tolerate a small amount of error without degrading their usability.
- The nature of some of the data sets and the analysis techniques is such that changes in a few data values will not affect the results.
- Relational data is different from multimedia data techniques developed for multimedia data cannot be directly used for watermarking relational data

Model


Alice
- relation R
- η tuples
- ω marked tuples
Steals Alice's data
Mallory
Attacker

Notations for reference

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

Detectability

- Alice should be able to detect her watermark by examining ω tuples from a suspicious database
- It is reasonable for Alice to get suspicious if her pattern is present in at least τ tuples (τ <= ω), where τ depends on ω and a preselected value α , called the significance level of the test.
- α : Determines how amenable the system is to false hits. That the probability that Alice will discover her watermark in a database relation not marked by her.

Robustness

- Watermarks should be robust against attacks to erase them.
- If Mallory changes ζ tuples of Alice's relation R . We say that the watermark is safe from erasure if the attack is not able to destroy the marks of at least τ tuples.

Incremental Updatability

- Having watermarked R , Alice should be able to update R in the future without destroying the watermark.
- The watermark values should only be recomputed for the added or modified tuples.

Imperceptibility

- The modifications caused by marks should not reduce the usefulness of the database

Blind System

- Watermark detection should neither require the knowledge of the original database nor the watermark

Key-Based System

- The watermarking system should assume that the method used for inserting a watermark is public.
- Defense must lie only in the choice of the private key.

Benign Updates and Malicious Attacks

- **Benign Updates**: Suppose Mallory has stolen Alice's data without realizing that it has been watermarked. Subsequently, Mallory may update the stolen data as he uses it. Watermarking should be such that Alice does not lose her watermark in the stolen data in spite of Mallory's updates.
- **Malicious Attacks**: Mallory may know that the data he has stolen contains a watermark but he may try to erase the watermark or try other means for claiming false ownership.

Message Authenticated Code

- A one-way hash function operates on an input message **M** of arbitrary length and returns a fixed length hash value **h**.

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

- r.P -- primary key attribute of tuple r.
- K -- private key which is only known to the owner.
- o represents concatenation.

Watermark Insertion Algorithm

// The private key $\mathcal{K}$ is known only to the owner of the database.
// The parameters γ, ν, and ξ are also private to the owner.

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

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

7) first_hash = $\mathcal{H}(\mathcal{K} \circ pk)$

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

Watermark Insertion Algorithm

- For erasing a watermark, therefore, the attacker will have to guess not only the tuples, but also the marked attribute within a tuple as well as the bit position.
- If a null attribute value is encountered while marking a tuple, we do not apply the mark to the null value, leaving it unchanged.

Watermark Detection Algorithm

// 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) **if** ($\mathcal{F}(s.P)$ mod γ equals 0) **then** // this tuple was marked
4) attribute_index $i = \mathcal{F}(s.P)$ mod ν // attribute A_i was marked
5) bit_index $j = \mathcal{F}(s.P)$ mod ξ // j^{th} bit was marked
6) totalcount = totalcount + 1
7) matchcount = matchcount + match($s.P$, $s.A_i$, j)

8) τ = threshold(totalcount, α) // see Section 4.2
9) **if** (matchcount $\geq \tau$) **then** *suspect piracy*

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

11) first_hash = $\mathcal{H}(K \circ pk)$

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

- Alice suspects that the relation S published by Mallory has been pirated from her relation R . The set of tuples and attributes in S can be a subset of R.
- Mallory does not drop the primary key attribute to preserve the usefulness of the data.
- The algorithm is probabilistic and only a certain minimum number of tuples have to contain matching marked bits.

“Robust and Blind Watermarking of Relational Database Systems”

Ali Al-Haj & Ashraf Odeh
PSUT, Jordan

2008

Introduction

Blindness: Watermark extraction should neither require the knowledge of the original un-watermarked database nor the watermark itself. 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.

Watermarking algorithm

1. Arrange the watermark image into m strings each of n bits length.
2. Divide the database logically into sub-sets of tuples. A sub-set has m tuples.
3. Embed the m short strings of the watermark image into each m-tuple sub-set.

Watermarking algorithm

4. Embed the n-bit binary string in the corresponding tuple of a sub-set as follows:

- Find the decimal equivalent of the string. Let the decimal equivalent be d
- Embed the decimal number d in a preselected nonnumeric, multi-word attribute by creating a double space after d words of the attribute

5. Embed the m short strings of the watermark image into each m-tuple sub-set.

6. Repeat steps 4 and 5 for each subset of the database under watermarking

Watermarking algorithm

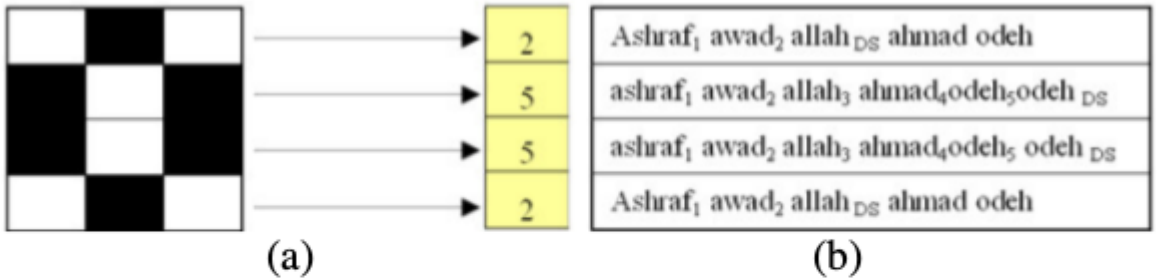


Fig. 1: (a): Binary image watermark and its decimal equivalent vector, (b): Watermarking example; where subscripts represent space number and DS corresponds to double space

Watermark extraction

1: Arrange the watermark image into m strings each of n bits length

2: Divide the database logically into sub-sets of tuples. A sub-set has m tuples

3: Embed the m short strings of the watermark image into each m-tuple sub-set.

Watermark extraction

4: Embed the n-bit binary string in the

corresponding tuple of a sub-set as follows

Find the decimal equivalent of the string and give it the symbol d

Embed the decimal number d in a preselected nonnumeric, multi-word attribute by creating a double space after d words of the attribute

5: Repeat step 4 for each tuple in the subset

6: Repeat steps 4 and 5 for each subset of the database under watermarking

END