3-2005

Fingerprinting relational databases: Schemes and specialities

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Vipin Swarup

Sushil Jajodia

DOI: https://doi.org/10.1109/TDSC.2005.12

Published in IEEE Transactions on Dependable and Secure Computing, 2005 March, Volume 2, Issue 1, Pages 34-45
https://doi.org/10.1109/TDSC.2005.12

Fingerprinting Relational Databases: Schemes and Specialties

Yingjiu Li, *Member*, *IEEE*, Vipin Swarup, and Sushil Jajodia, *Senior Member*, *IEEE*

Abstract—In this paper, we present a technique for fingerprinting relational data by extending Agrawal et al.'s watermarking scheme. The primary new capability provided by our scheme is that, under reasonable assumptions, it can embed and detect arbitrary bit-string marks in relations. This capability, which is not provided by prior techniques, permits our scheme to be used as a fingerprinting scheme. We then present quantitative models of the robustness properties of our scheme. These models demonstrate that fingerprints embedded by our scheme are detectable and robust against a wide variety of attacks including collusion attacks.

Index Terms—Fingerprint, relational database, robustness, collusion attack.

1 Introduction

Fingerprinting is a class of information hiding techniques that insert digital marks into data with the purpose of identifying the recipients who have been provided data. *Watermarking* is another class of information hiding techniques whose purpose is to identify the sources of data. Both techniques can help protect data from piracy, which has become a severe threat to database applications [25].

Consider a generic scenario where merchants (or owners, we use "owners" and "merchants" interchangeably) sell digital data to buyers. Some dishonest buyers (called traitors) may redistribute the data to others without permission from the merchants. A merchant may use a watermarking scheme to embed a merchant-specific mark into her data; she can subsequently detect the mark in pirated data and use the detection process to assert ownership of the data. She may use a fingerprinting scheme to embed a buyer-specific mark into a data copy provided to a buyer; she can subsequently detect the mark in pirated data and use the mark to identify the traitor who distributed the data.

Watermarking and fingerprinting have different goals and, hence, require different schemes. Watermarking aims to identify a data owner and, hence, is subject to attacks where a pirate claims ownership of the data or weakens a merchant's claims. In contrast, fingerprinting aims to identify a traitor and is subject to attacks that cause an innocent principal (or no principal) to be identified as a traitor.

- *Y. Li is with the School of Information Systems, Singapore Management University, 469 Bukit Timah Road, Singapore 259756. E-mail: yjli@smu.edu.sg.*
- *V. Swarup is with The MITRE Corporation, 7515 Colshire Drive, McLean, VA 22102. E-mail: swarup@mitre.org.*
- *S. Jajodia is with the Center for Secure Information Systems, George Mason University, Mail Stop 4A4, 4400 University Drive, Fairfax, VA 22030. E-mail: jajodia@gmu.edu.*

Manuscript received 5 Mar. 2004; revised 4 Dec. 2004; accepted 22 Feb. 2005; published online 4 Apr. 2005.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0043-0304.

Watermarking and fingerprinting have been studied extensively in the context of multimedia data (e.g., [5], [6], [11], [12]). However, little work has been done on fingerprinting in the context of relational data. Very recently, research has targeted the watermarking of relational databases [1], [9], [23], [24]. In [1], a unique scheme (AK scheme for convenience) is presented for embedding watermarks within numeric attributes of relations. It assumes that merchants and buyers can tolerate a small amount of errors in those attributes, but that introducing many more errors will reduce the value of the data substantially. For example, if an attribute contains numeric values with five decimal places, then a buyer may accept that 0.1 percent of the tuples have arbitrary fractional errors as long as the rest are accurate up to five decimal places.

In the AK scheme, each watermark is represented by a unique secret key $\mathcal{K}$. This key must be used during both watermark insertion and detection. In watermark insertion, the key determines (pseudorandomly) multiple mark bits and the positions where the mark bits are embedded in a database relation; the detection algorithm merely tests whether or not a specific key was used to mark the relation. Since the mark bits are determined by the key, they represent a single identity who possesses the key. This is inadequate for many marking applications, e.g., for fingerprinting, where a merchant must have the ability to embed and detect distinct marks or identities in separate copies of a relation. In this paper, we will present a marking scheme that permits an arbitrary mark bitstring to be embedded in a relation using a single secret key. The mark bitstring can be used to represent different buyers who purchase the database relation. Our detection algorithm tests whether a key was used to mark a relation and, if so, it returns the actual mark bitstring that was embedded.

Our marking scheme can be used for both watermarking (the same bitstring is embedded and detected) and fingerprinting (different bitstrings are embedded and detected). When used for watermarking, our scheme has a significant impact on the robustness due to the change of

TABLE 1
Notation

P:	primary key attribute	ν:	number of attributes
η:	number of tuples	$1/\gamma$:	fraction of tuples used in fingerprinting
$\mathcal{K}$:	secret key	$\mathcal{S}$:	pseudo-random sequence generator
L:	length of fingerprint	N:	number of buyers

embedding marks. In a separate paper [14], we have proven that our scheme is more robust than the AK scheme in the sense that it provides an upper bound for the probability that a valid watermark is detected from pirated data, and an upper bound for the probability that a fictitious secret key is discovered from pirated data. A further comparison between watermarking and fingerprinting schemes is given in Section 3.

The focus of this paper is to apply such a marking scheme to fingerprinting relational databases and present a rigorous analysis on the specialties in fingerprinting: 1) We define several measures for the robustness properties of fingerprinting schemes. 2) We present detailed quantitative models of these measures for our fingerprinting scheme. 3) We demonstrate the robustness of our scheme under collusion attacks (which are unique to fingerprinting). This body of new quantitative analysis is the primary contribution of this paper.

The rest of the paper is organized as follows: Section 2 presents our fingerprinting scheme. Section 3 compares our scheme with existing watermarking schemes. Section 4 presents a detailed analysis of the specialties of our fingerprinting scheme. Section 5 addresses the collusion problem in fingerprinting relational databases. Section 6 discusses the extension to our work. Section 7 reviews related work. Finally, Section 8 summarizes the paper and suggests future directions.

2 A Fingerprinting Scheme for Relational Data

Our fingerprinting scheme is developed by extending the AK scheme, which was proposed by Agrawal et al. for watermarking relational databases [1]. We first present our scheme and then clarify the relationship between our scheme and AK scheme. Table 1 gives the notation that will be used in our scheme.

Consider a database relation R with primary key P and ν numerical attributes $A_0, \ldots, A_{\nu-1}$. Assume that it is acceptable to change one of ξ least significant bits in a small number of numeric values. The relation has η tuples and a fraction $1/\gamma$ of them will be used for fingerprinting.

The owner of R has a secret key $\mathcal{K}$. A cryptographic pseudorandom sequence generator [22] $\mathcal{S}$ is used to select tuples, attributes, bits, and decide how to change the bits in fingerprinting. Such a pseudorandom sequence generator produces a sequence of numbers from an initial seed. Without the knowledge of the seed, it is computationally infeasible to compute the next number in the sequence. Different seeds lead to different sequences. For each tuple r, $\mathcal{S}$ is seeded with the primary key $r.P$ concatenated with the secret key. Let $\mathcal{S}_i(\mathcal{K}, r.P)$ denote the ith number in the sequence generated by $\mathcal{S}$. The values of $\mathcal{S}_i(\mathcal{K}, r.P)$ are uniformly distributed for different primary key values.

2.1 Fingerprint Codes

It is assumed that there are N buyers and that all fingerprints are unique binary codewords (i.e., strings) and have the same length L ($L > \ln N$). In general, any binary codewords of length L can serve as fingerprints. However, this requires storing all buyer-fingerprint pairs in a database for detecting purpose. Not only does this requirement violate the key-based property, but also introduces management overhead for keeping the database secret (it is critical to keep all fingerprints secret to thwart a collusion attack); some security mechanisms such as access control and encryption may need to be enforced so as to protect the database.

To avoid this, let the merchant generate each buyer's fingerprint $\mathcal{F}$ from the merchant's secret key $\mathcal{K}$ and the buyer's series number i (which can be public) using a cryptographic hash function[1] $\mathcal{H}$,

$$\mathcal{F}(\mathcal{K}, i) = (f_0, \ldots, f_{L-1}) = \mathcal{H}(\mathcal{K}|i),$$

where "|" indicates concatenation. There is no need to store the fingerprints which can be computed on the fly during fingerprint insertion or detection.

The secret key and fingerprinting codes should be long enough to thwart exhaustive search and various types of attacks. We assume $L \gg \ln N$. L can be, for example, 64, 128, or 160 bits. In most cases, a 128-bit secret key and 64-bit fingerprinting codes should be good enough. Note that the fingerprinting codes described in this section are not collusion resistant and that the collusion issue will be addressed in Section 5 using Boneh and Shaw's fingerprinting codes.

2.2 Algorithms

Our fingerprinting scheme consists of two algorithms, fingerprint insertion and fingerprint detection. The fingerprint insertion algorithm is shown in Fig. 1. The algorithm inserts the fingerprint of buyer n into relation R. For each tuple r of R, the algorithm seeds the sequence generator $\mathcal{S}$ with the concatenation of the secret key $\mathcal{K}$ and the primary key $r.P$ of the tuple. If the first sequence number $\mathcal{S}_1(\mathcal{K}, r.P) \bmod \gamma = 0$, the tuple is selected. Therefore, on average, one out of γ tuples are selected. For each selected

1. Using a standard hash function such as MD5, $\mathcal{H}(\mathcal{K}|i)$ may not have the exact selected length L. We can either truncate it or concatenate multiple copies of it to obtain an L-bit fingerprint.

// insert fingerprint to relation R with scheme $(P, A_0, \ldots, A_{\nu-1})$

fingerprint of buyer n: $\mathcal{F}(\mathcal{K}, n) = \mathcal{H}(\mathcal{K}|n)$ // $\mathcal{F}(\mathcal{K}, n) = (f_0, \ldots, f_{L-1})$

foreach tuple $r \in R$ **do**

  if ($\mathcal{S}_1(\mathcal{K}, r.P)$ mod γ equals 0) **then** // mark this tuple

    attribute_index $i = \mathcal{S}_2(\mathcal{K}, r.P)$ mod ν // mark attribute A_i

    bit_index $j = \mathcal{S}_3(\mathcal{K}, r.P)$ mod ξ // mark least significant bit j

    mask_bit $x = 0$ if $\mathcal{S}_4(\mathcal{K}, r.P)$ is even; $x = 1$ otherwise

    fingerprint_index $l = \mathcal{S}_5(\mathcal{K}, r.P)$ mod L

    fingerprint_bit $f = f_l$ // select fingerprint bit l

    mark_bit $m = x \oplus f$

    set least significant bit j of $r.A_i$ to m.

return R

Fig. 1. Fingerprint Insertion Algorithm.

tuple, the algorithm selects exactly one attribute; in particular, it selects attribute i if $\mathcal{S}_2(\mathcal{K}, r.P) \bmod \nu = i$. Similarly, it selects least significant bit j from the selected attribute if $\mathcal{S}_3(\mathcal{K}, r.P) \bmod \xi = j$. On the other hand, the algorithm computes a mask bit x according to $\mathcal{S}_4(\mathcal{K}, r.P)$; if $\mathcal{S}_4(\mathcal{K}, r.P)$ is even, then $x = 0$ else $x = 1$. It also selects a fingerprint bit f_l if $\mathcal{S}_5(\mathcal{K}, r.P) \ \ mod\ L = l$. Finally, the algorithm XOR's the fingerprint bit with the mask bit and assigns the selected bit with the XOR'ing result. The purpose of using mask bits is to hide the distribution of fingerprint bits.

Note that all selections and computations in fingerprint insertion are based on $\mathcal{S}_i(\mathcal{K}, r.P)$ which are uniformly distributed. On average, η/γ bits (or tuples) are used to embed a fingerprint, and each fingerprint bit f_l is embedded $\eta/(\gamma L)$ times. Also, note that the secret key is involved in every step of the process. Without knowing the secret key and without comparing multiple fingerprint copies, buyers are prevented from knowing where the fingerprint is embedded.

In fingerprint detection, a merchant of database relation R would like to determine whether another relation R' was pirated from R and, if so, identify the traitor who distributed R' without authorization. If R' is pirated, the algorithm assumes that the primary key attribute values as well as the order among marked attributes have not changed (or else can be recovered). Note that R' may consist of only a subset of original tuples; it may include some additional tuples that were not in R, and some bit values in R' could have been changed by the traitor before detection.

The fingerprint detection algorithm is shown in Fig. 2. The algorithm initiates a fingerprint template $\mathcal{F} = (f_0, \ldots, f_{L-1})$ as $(?, \ldots, ?)$, where "?" indicates that a mark bit is in an unreadable state (such a symbol is also used in other fingerprinting papers, such as [3], [21]). It then locates the marked bits exactly as the insertion algorithm does. From each marked bit, the algorithm extracts a fingerprint bit f_l by XOR'ing the bit value with a computed mask bit. If the marked bit has been changed by the traitor, the extracted f_l may not match its original value. The algorithm uses two counting variables $count[l][0]$ and $count[l][1]$ to indicate the number of times that f_l is extracted to be 0 and 1, respectively. After all marked bits are checked, the algorithm assigns 0 (or 1, respectively) to f_l if the ratio $count[i][0]/(count[i][0] + count[i][1])$ (or $count[i][1]/(count[i][0] + count[i][1])$, respectively) is greater than τ, where $\tau \in [0.5, 1)$ is a real parameter that is related to the assurance of the detection process.

A traitor is detected in a subroutine *detect* if the recovered fingerprint template $\mathcal{F} = (f_0, \ldots, f_{L-1})$ matches one of the N buyers' fingerprints, which is computed in the same manner as the insertion algorithm (this can be either calculated on the fly, as shown in Fig. 2, or calculated before fingerprint detection). There are various methods on how to match a binary string against a set of strings efficiently (e.g., through a Bloom filter [2]). Any of such methods can be incorporated into our detection algorithm. We simply use the exact match in the algorithm.

3 Comparison with Watermarking Schemes

In this section, we compare our fingerprinting scheme with the AK scheme as well as other watermarking schemes in regards to embedding methods, assumptions, properties, errors, and robustness measures.

3.1 Embedding Methods

The original form of the AK scheme *physically* embeds and detects ω ($\omega = \frac{\eta}{\gamma}$) mark bits, where each mark bit is computed the same way as the mask bit in our scheme. Since the mark bits (and where they are embedded) are determined by the secret key, they represent a single identity who possesses the key. This is inadequate for fingerprinting applications where a merchant must have the ability to embed and detect distinct marks or identities in separate copies of a relation. Our scheme generalizes

// detect fingerprint from relation R' with scheme $(P, A_0, \ldots, A_{\nu-1})$
// $\mathcal{K}, \gamma, \nu, \xi$ are the same as in fingerprint insertion

// initiate fingerprint template and counts
  fingerprint template $\mathcal{F} = (f_0, \ldots, f_{L-1}) = (?, \ldots, ?)$
    // '?' represents an unknown value
  foreach $i = 0$ to $L - 1$ **do** $count[i][0] = count[i][1] = 0$
    // $count[i][0]$, $count[i][1]$ are votes for f_i to be 0 and 1 respectively

// scan all tuples and obtain counts for each fingerprint bit
  foreach tuple $r \in R'$ **do**
    if $(\mathcal{S}_1(\mathcal{K}, r.P)$ mod γ equals 0) **then** // this tuple was marked
      attribute_index $i = \mathcal{S}_2(\mathcal{K}, r.P)$ mod ν // attribute A_i was marked
      bit_index $j = \mathcal{S}_3(\mathcal{K}, r.P)$ mod ξ // bit j was marked
      if least significant bit j of $r.A_i$ does not exist, **then**
        skip to the next tuple
      mark_bit $m =$ least significant bit j of $r.A_i$
      mask_bit $x = 0$ if $\mathcal{S}_4(\mathcal{K}, r.P)$ is even; $x = 1$ otherwise
      fingerprint_bit $f = m \oplus x$
      fingerprint_index $i = \mathcal{S}_5(\mathcal{K}, r.P)$ mod L
      $count[i][f] = count[i][f] + 1$ // update the votes

// recover fingerprint
  foreach $i = 0$ to $L - 1$ **do**
    if $count[i][0] + count[i][1] = 0$ **then return** *none suspected*
    $f_i = 0$ if $count[i][0]/(count[i][0] + count[i][1]) > \tau$;
    $f_i = 1$ if $count[i][1]/(count[i][0] + count[i][1]) > \tau$;
    return *none suspected* otherwise
  $\mathcal{F} = (f_0, \ldots f_{L-1})$

// determine a traitor
  buyer $n = detect\ (\mathcal{F}, \mathcal{K}, L, N)$ // detect a traitor based on template $\mathcal{F}$
  if $n \geq 0$ **then return** *buyer n is the traitor*
  else return *none suspected*

// subroutine: detect a traitor
  detect (template $\mathcal{F}$, secret key $\mathcal{K}$, fingerprint length L, number of buyers N)
    foreach buyer $n = 0$ to $N - 1$ **do**
      $\mathcal{F}' = \mathcal{H}(\mathcal{K}|n)$
      if $\mathcal{F}$ matches $\mathcal{F}'$ **then return** n
    return -1

Fig. 2. Fingerprint Detection Algorithm (note: this algorithm is not collusion resistant).

AK scheme in a simple way based on a "logical view" of the AK scheme. In this view, the AK scheme *logically* embeds and detects a single bit ω times through XOR'ing mask bits:

- In watermark insertion, embedding a mark bit x is equivalent to embedding "fingerprint bit" "0" through mask bit x ($0 \oplus x = x$).
- In watermark detection, extracting a mark bit x is equivalent to extracting "fingerprint bit" "0" through mask bit x ($x \oplus x = 0$).

From this point of view, the AK scheme can be considered as a special case of our scheme where $L = 1$ and $f_0 = 0$.

There could be other ways to generalize the AK scheme for embedding fingerprints. For instance, if each fingerprint is mapped to a different secret key, the merchant may use the AK scheme with the appropriate secret key to embed a particular fingerprint. However, fingerprint detection is inefficient and unscalable because it may require checking all possible keys before reaching a decision. In addition, this solution is vulnerable to collusion attack where a group of buyers collaboratively compare their data copies, identify differences of embedded marks, and modify their fingerprints. Another extension is to partition the data and then use the AK scheme to embed each fingerprint bit into a unique partition. However, this scheme is vulnerable to those attacks that delete a large portion of tuples. For example, in the case that half of the partition blocks are deleted in an attack, half of the fingerprint bits will not be detected anyway.

People have proposed other methods so as to directly embed a bitstring to relational data [16], [24], [27]. In [24], an arbitrary bit is embedded into a selected subset of data values by changing the distribution of the values. The selection of subsets is based on a secret sorting, which could be expensive for large data sets. In [16], a database relation is transformed into a two-dimensional image into which a bitstring is embedded, while, in [27], a bitstring is embedded by adding or not adding a set of extra tuples called stealth tuples. We mention that a naive application of these methods to embedding fingerprints is not secure or efficient. We shall make this clearer in Section 6.3 after studying collusion attacks.

3.2 Assumptions and Properties

Compared to the AK scheme, our fingerprint scheme shares the same set of assumptions, including 1) database users can tolerate the errors that are caused by changing one of ξ least significant bits in a small number of numerical values (the number is controlled by parameter γ), 2) a database relation has a primary key attribute which does not change or else can be recovered, and 3) the order of marked attributes does not change or else can be recovered. Note that the latter two assumptions are not critical to this approach and may be dropped with suitable extensions (see [13], for instance).

Our fingerprinting scheme also inherits the same set of properties from the AK scheme including: 1) Key-based: The secret key is involved in every step of the scheme for embedding and detecting a fingerprint. 2) Blind: It is not required to have the original database or any fingerprint involved in fingerprint detection. 3) Incrementally updatable: Each tuple is assumed to have a unique primary key, based on which it is processed independently.

3.3 Errors

Compared to the AK-scheme, our fingerprint scheme uses the same procedure to locate marked bits in data. The procedure is determined by the secret key, the primary key attribute, and the parameters γ, ν, ξ. With the same set of parameters, both schemes locate the same marked bits for embedding fingerprint or watermark. The only difference is that, in the AK scheme, mark bits (i.e., mask bits in our scheme) x are embedded directly, while, in our scheme, the same bits x are XOR'ed with corresponding fingerprint bits f before being embedded (i.e., $x \oplus f$ are embedded). Because x are determined pseudorandomly by $\mathcal{S}_4(\mathcal{K}, r.P)$, both schemes modify the underlying data with the same probability. Therefore, the errors introduced by our scheme are statistically the same as by watermarking scheme.

We do not repeat the error analysis which has been done on the AK scheme [1]. Rather, we summarize that the errors can be controlled to be minuscule by appropriate selection of parameters γ, ν, ξ. The rationale behind this is as follows:

- The number of attribute values modified during fingerprint insertion is small. In a database with η tuples, on average, η/γ tuples are selected for marking. In each of these tuples, a single bit from a single attribute is replaced by a mark value. Since the mark values are selected pseudorandomly, they will equal the unmarked bit values about half the time and, so, will not modify those bits.
- The errors introduced in attribute values are bounded. When an attribute is marked, only one of its ξ least significant bits is replaced by a mark value. This restricts the changes in attribute values caused by marking. A merchant can trade off γ against ξ to control the accuracy of the marked data. That is, the merchant can reduce the number of marked attributes by increasing the errors introduced in each marked attribute.

It should be noted that, no matter how small the errors are, the fingerprinting process does alter the underlying data as well as its structural properties. For example, if one considers a natural join operation in which equalities are specified on k fingerprinted attributes, the probability that a tuple is missed in the result of the operation is at most $1 - (1 - 1/(2\gamma\nu))^k$. Therefore, one should be careful about the intended use of the fingerprinted data. As indicated in [1] and the Appendix, the errors introduced by watermarking or fingerprinting are small enough (in terms of attribute statistics such as mean and variance), and some data mining task such as classification is not affected. One may also apply so-called "on-the-fly quality assessment" [24] so as to make sure that the usability of data is not affected by the fingerprinting process in practice.

3.4 Robustness Measures

Fingerprinting schemes should be robust against benign database operations and malicious attacks that may destroy or modify embedded fingerprints. It should be hard for attacks and benign updates to erase embedded fingerprints, to modify embedded fingerprints so that an innocent buyer is implicated as a traitor, or to modify unmarked data so that the fingerprinting scheme detects a valid but incorrect fingerprint in the modified data.

As indicated in [1], the embedded marks can always be destroyed by making substantial modifications to marked data. Therefore, the robustness of our scheme is considered relative to the data modifications made by attacks and updates.

We analyze our fingerprinting scheme using the following robustness measures. We say that a binary string is a *valid* fingerprint if it is of the form $\mathcal{F}(\mathcal{K}, n)$ $(0 \leq n < N)$. A detected fingerprint is *incorrect* if it is not the one that was used in fingerprint insertion.

- *Misdiagnosis false hit (fh^D)*: The probability of detecting a valid fingerprint from data that has not been fingerprinted.
- *Misattribution false hit (fh^A)*: The probability of detecting an incorrect but valid fingerprint from fingerprinted data.
- *False negative (fn)*: The probability of detecting no valid fingerprint from fingerprinted data.
- *False miss (fm)*: The probability of failing to detect an embedded fingerprint correctly. False miss rate is the sum of the false negative and misattribution false hit rates, that is, $fm = fh^A + fn$.

These robustness measures are subtly different from those used for watermarking [1], [14]. In watermarking, there are only two types of false detection rates: detecting a valid watermark from unmarked data (false hit) and detecting no watermark from marked data (false miss). The existence of multiple buyer's fingerprints demands two types of false detection rates from marked data: Either no fingerprint is detected (false negative) or a valid but incorrect fingerprint is detected (misattribution false hit). Roughly speaking, false miss (sum of false negative and misattribution false hit) and misdiagnosis false hit in fingerprinting setting are the counterparts of false miss and false hit in watermarking setting.

3.4.1 Note

The fingerprinting algorithms presented in Section 2.2 are similar to that we developed for watermarking [14] except that 1) a unique fingerprint code is generated for each buyer, and 2) an additional procedure *detect* is required for detecting a traitor.

So far, we have not studied the collusion attack, which is specific to fingerprinting. We shall show that a collusion resistant scheme has to combine a watermarking code [14] and a collusion-resistant code [3]; otherwise, it could result in 100 percent misdiagnosis false hit rate (see Section 5.1). Therefore, a direct extension of the watermarking scheme for fingerprinting is not acceptable.

The analysis in the previous paper [14] focuses on the significant impact of using multiple watermark bits on the false hit and false miss rates, while this paper concentrates on the N different buyer issues, such as misattribution false hit, invertibility attack, and collusion attack. Uniquely, this paper studies the specialties of a collusion secure fingerprinting scheme with regard to problems, algorithm, analysis and experiment.

The overlap between this paper and the previous paper includes the initial embedding method, analytical model (cumulative binomials), and data errors. To make this paper complete, this part of the work is repeated, but it is kept minimal as consistent with its title.

4 Analysis

We analyze the robustness of our scheme under a range of representative attacks. Among those attacks, some of them are common to both fingerprinting and watermarking, while collusion attacks are unique to fingerprinting. Our analytical approach for the common part of attacks is similar to that used for watermarking schemes [1], [14]. However, as mentioned above, we use a different set of robustness measures (i.e., misdiagnosis false hit, misattribution false hit, and false negative). We brief our results on this in this section and pay more attention to the collusion attacks in the next section.

We will use the following notation: Let $b(k; n, p) = \binom{n}{k} p^k q^{n-k}$ be the binomial distribution function which gives the probability of obtaining *exactly* k successes out of n Bernoulli trials, where the result of each Bernoulli trial is true with probability p and false with probability $q = 1 - p$. Let $B(k; n, p) = \sum_{i=k+1}^{n} b(i; n, p)$ be the binomial distribution survival function which returns the probability of having *more than* k successes in n independent Bernoulli trials. Function $B(k; n, p)$ is monotonic decreasing with k. In particular, $B(k; n, p) \leq 0.5$ if $p = 0.5$ and $k \geq \frac{n}{2}$.

4.1 Misdiagnosis False Hit

It is clear that, in the absence of malicious attacks or benign updates, our fingerprinting scheme detects the embedded fingerprint from pirated data. However, the scheme may extract a fingerprint (purely by chance) from unmarked data. We investigate the misdiagnosis false hit in such a case.

If the detection algorithm is applied to unmarked data, it may return some binary string $(f_0, \ldots, f_{L-1})$ as a potential fingerprint. Let f_i be extracted from data ω_i times $(\omega_i > 0)$. Due to the use of pseudorandom mask bits, each time f_i is extracted, it will be extracted as zero or one with probability 0.5, which is modeled as an independent Bernoulli trial. After all data is processed, f_i is detected to be zero or one with the same probability $B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5)$. The algorithm detects a binary string as a potential fingerprint with probability $\Pi_{i=0}^{L-1} 2B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5)$, where the factor 2 means that each bit could be either zero or one. Now, the N valid fingerprints are selected pseudorandomly from the 2^L possible binary strings. Thus, the probability that the binary string is a valid fingerprint is $\frac{N}{2^L}$. The overall misdiagnosis false hit rate is

$$fh^D = \frac{N}{2^L}\Pi_{i=0}^{L-1} 2B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5) = N \cdot \Pi_{i=0}^{L-1} B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5).$$

The misdiagnosis false hit has upper bound $\frac{N}{2^L}$ since $B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5) \leq 0.5$ when $\tau \in [0.5, 1)$. Note that the upper bound is tight in the case that all ω_i are odd and $\tau = 0.5$. The misdiagnosis false hit rate can be reduced exponentially by increasing L ($fh^D \simeq 0$ if $L \gg \log N$); it can also be reduced by increasing τ. The upper bound is independent of the size of the database relation.

4.2 Bit-Flipping Attack

In the presence of various attacks, a fingerprinting scheme may not be able to detect the embedded fingerprint from pirated data (false negative); worse, it may extract an innocent buyer's fingerprint (misattribution false hit).

First, consider a bit-flipping attack where an attacker randomly selects some bits and toggles their values [1], [14]. Assume that the attack toggles each least significant bit with probability p. Let $q = 1 - p$. Also assume that less than half of the fingerprintable bits are flipped (i.e., $p \leq 0.5$); otherwise, fingerprint detection can be applied to transformed data by flipping each fingerprintable bit back. The flipping of a bit is modeled as an independent Bernoulli trial with probability p of success and q of failure.

A bit flipping attack does not change the size of data. Assume that each fingerprint bit f_i is embedded $\omega_i > 0$ times and will be extracted exactly the same number of times. For the detection algorithm to fail to extract a correct fingerprint bit, at least $(1-\tau)\omega_i$ embedded bits that correspond to the fingerprint bit must be toggled (or, equivalently, more than $\omega_i - \lfloor \tau\omega_i \rfloor - 1$ bits are toggled). Thus, the probability that the fingerprint bit is detected incorrectly is $B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p)$. The probability that the entire fingerprint is detected incorrectly (i.e., the false miss rate) is

$$fm = 1 - \Pi_{i=0}^{L-1}(1 - B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p)).$$

Now, consider the misattribution false hit fh^A. For the detection algorithm to extract a binary bit for fingerprint bit f_i, either at most $\omega_i - \lfloor \tau\omega_i \rfloor - 1$ of its embedded bits are toggled, or more than $\lfloor \tau\omega_i \rfloor$ of its embedded bits are toggled. If the detection algorithm extracts a binary string, the probability that the binary string is a valid but "innocent" fingerprint is $\frac{N-1}{2^L}$. Therefore,

$$fh^A = \frac{N-1}{2^L}\Pi_{i=0}^{L-1}(1 - B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p) + B(\lfloor \tau\omega_i \rfloor; \omega_i, p)) \leq \frac{N-1}{2^L}.$$

The misattribution false hit rate has an upper bound $\frac{N-1}{2^L}$ since $B(\lfloor \tau\omega_i \rfloor; \omega_i, p) \leq B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p)$ when $\tau \in [0.5, 1)$. The upper bound is tight in the case that all ω_i are odd and $\tau = 0.5$. It is straightforward to get the false negative

$$fn = 1 - \Pi_{i=0}^{L-1}(1 - B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p)) - \frac{N-1}{2^L}\Pi_{i=0}^{L-1}(1 - B(\omega_i - \lfloor \tau\omega_i \rfloor - 1; \omega_i, p) + B(\lfloor \tau\omega_i \rfloor; \omega_i, p)).$$

4.3 Subset Attack and Superset Attack

Assume that a pirate examines each tuple independently and selects it with probability p for inclusion in the pirated relation. We call this subset attack.

A subset attack cannot succeed unless it deletes all the embedded bits for at least one fingerprint bit. Assume that each fingerprint bit f_i is embedded ω_i times. The probability that all the embedded bits for f_i are deleted is $B(\omega_i - 1; \omega_i, 1-p) = (1-p)^{\omega_i}$. Then, the false miss rate is $fm = 1 - \Pi_{i=0}^{L-1}(1 - (1-p)^{\omega_i})$. Since the subset attack alone does not change any tuples that are included in the pirated data, the false negative rate is the same as the false miss rate, and the misattribution false hit rate is zero.

A dual attack of a subset attack is a superset attack. In a superset attack, an attacker takes a pirated relation and mixes it with tuples from other sources to create a relation. Note that the mix-and-match attack discussed in [1] can be considered as a combination of subset attack and superset attack. Assume that fingerprint bit f_i is embedded ω_i times in the original data, and that it is extracted ω_i' times from the additional tuples. Then, the probability that this fingerprint bit is destroyed in a superset attack is $B(\omega_i + \omega_i' - \lfloor (\omega_i + \omega_i')\tau \rfloor - 1; \omega_i', 0.5)$. It is fairly straightforward to derive various robustness measures for this attack as for subset attack.

4.4 Invertibility Attack

Consider a scenario where a pirate randomly selects a secret key and runs the fingerprint detection algorithm on a pirated relation. The "key" may cause the fingerprint detection algorithm to extract a valid fingerprint from the pirated relation. This type of attacks is referred to as an invertibility attack [7]. If such an attack succeeds, a pirate can use the discovered key to claim legitimate ownership of the data. Alternately, a pirate can claim innocence by claiming that the merchant used this attack to obtain evidence of piracy.

The probability that a tried key will lead to a valid fingerprint being detected is given by

$$\max\left(\frac{1}{2^{|\mathcal{K}|}}, N \cdot \Pi_{i=0}^{L-1} B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5)\right),$$

where ω_i is the number of times that fingerprint bit i is extracted from data, the first term $\frac{1}{2^{|\mathcal{K}|}}$ is the probability that the tried key is the real secret key (assume that the length of the secret key is fixed and public), and the second term is the misdiagnosis false hit (detecting fingerprint from pirated data using an incorrect key is modeled the same as detecting fingerprint from unmarked data using the correct key). An attacker can choose the parameters γ, L, τ, and N to increase his probability shown in the second term (note $\omega_i \simeq \frac{\eta}{\gamma L}$). In particular, if he selects $\tau = 0.5$, this may reduce to $\frac{N}{2^L}$. The attacker can then select N and L to increase his likelihood of success.

Thwarting this attack requires that L be much larger than $logN$ (e.g., $L - \log N \geq 60$) and that $\mathcal{K}$ be long enough (e.g., $|\mathcal{K}| = 128$). This requirement can be enforced by convention. Note that an alternate convention might be to require τ to be greater than 0.5; however, an attacker may get around that

convention by first reducing ω_i (e.g., via a subset attack) before launching an invertibility attack.

There are some other types of attacks, such as additive attack, by which a pirate inserts another fingerprint before distributing a pirated database, and combination attack, by which an attacker launches multiple attacks on the same data. These attacks can be analyzed by similar approaches as used in watermarking [1], [14].

5 Collusion

Fingerprinting schemes are susceptible to collusion attacks by coalitions with access to multiple fingerprinted copies of the same relation but with different embedded fingerprints. Members of a coalition may be able to create a useful data copy that does not implicate any member of the coalition. During fingerprint detection, either the copy may yield the fingerprint of an innocent buyer or it may not yield a valid fingerprint. The collusion attack is specific to fingerprinting and there is no collusion attack in watermarking setting.

5.1 Problem

The fingerprinting scheme of Section 2 is not secure against collusion attacks. For example, suppose that three fingerprints $(1,1,0)$, $(1,0,1)$, and $(0,1,1)$ are embedded in three copies (or instances) of the same database relation using the same secret key and parameters. A coalition which has all three copies can compare the copies and replace each bit position where the copies differ with the value shared by a majority of the copies (note that fingerprint bit i is embedded to the same positions in different copies). It is easy to see that the new copy will have an embedded binary string $(1,1,1)$. The coalition can also replace each identified bit with a random value, possibly producing a binary string $(1,0,0)$. These binary strings do not necessarily identify any members of the coalition as a traitor.

Collusion resistent fingerprinting codes have been studied extensively in the literature of cryptography (e.g., [3], [10], [15], [17], [18], [19], [26]). A well-known fingerprinting code, which we call BoSh code for short, was proposed by Boneh and Shaw [3]. The basic idea is to use information that a coalition cannot detect to trace one of the traitors. BoSh code is designed to be c-secure with ϵ-error as it enables the capture of a member of a coalition of at most c members with probability at least $1-\epsilon$. Integer c and real ϵ in $[0,1]$ are two parameters in the design of BoSh code. Increasing c or reducing ϵ results in longer BoSh codes.

The effectiveness of BoSh code depends on the "Marking Assumption," which states that colluding buyers can detect a specific fingerprint bit f_i if it differs between their copies; otherwise, a fingerprint bit cannot be detected [3]. Our fingerprinting scheme satisfies this assumption as each fingerprint bit f_i is embedded to the same positions in all copies of data. Our scheme is also independent of the composition of fingerprint codewords; thus, one can easily incorporate BoSh codes as buyer fingerprints into our scheme. However, this is inadequate for two reasons:

- The tracing algorithm proposed in [3] returns exactly one "traitor" no matter what the input is (the assumed input in [3] is pirated data under collusion attacks). If the input is a pirated data copy generated by a coalition, the error rate (i.e., misattribution false hit) is less than ϵ by design. However, if the input is unmarked data, then the misdiagnosis false hit rate will be 100 percent, which is unacceptable.
- The generation of BoSh codes requires recording of 1) a secret, random (outer) code for each buyer and 2) a secret, random permutation for all buyers. The owner must keep detailed records binding fingerprints to buyers for detection purpose. This requirement violates the key-based property.

5.2 Solution

Our solution is to partition each fingerprint $\mathcal{F}$ of L bits into two parts: The first part, $\mathcal{F}_1$ of L_1 bits, is used as a watermark, while the second part, $\mathcal{F}_2$ of L_2 bits, is used as a fingerprint. Thus, $L = L_1 + L_2$ and $\mathcal{F} = \mathcal{F}_1|\mathcal{F}_2$.

The watermark part $\mathcal{F}_1$ consists of multiple bits (thus, it is different from that in the AK scheme), and it is the same for all data buyers. It is computed directly from the secret key using a hash function $\mathcal{H}(\mathcal{K})$ and using truncation or concatenation to obtain a bit string of length L_1.

The fingerprint part $\mathcal{F}_2$ is generated using the BoSh code [3]. Let $a = 2c$, $b = 2c\log\frac{2N}{\epsilon}$, and $d = 2a^2\log\frac{4ab}{\epsilon}$. The BoSh code for buyer n is composed by concatenating b code words selected from a common "inner code" $\Gamma_0(a,d)$ according to a buyer-specific "outer code" O. The inner code $\Gamma_0(a,d)$ consists of a code words, each of which is a binary string of length $(a-1)d$. The outer code $O = (o_0, \ldots o_{b-1})$ is an integer vector, where each element $0 \leq o_i < a$ indicates which code word in the inner code is used in concatenation. The length of the fingerprint part $\mathcal{F}_2$ is thus $L_2 = b(a-1)d$. In Boneh and Shaw's work [3], a fixed, random permutation is applied on the fingerprint before use. The inner code is public, while the outer code and the permutation must be kept hidden from all buyers. The purpose of keeping the permutation hidden is to hide the information of which mark bit encodes which fingerprint bit [3]. Our insertion algorithm already has this property. Even if a buyer can detect a mark bit in a database tuple, he cannot tell which fingerprint bit it encodes (without knowledge of the secret key). Therefore, the permutation is not needed in our algorithm.

Another point is that the merchant does not need to remember the secret outer code for each buyer either. Instead of generating the outer code for each buyer randomly [3], we generate it using the secret key and the buyer's series number n pseudorandomly. We generate $o_i = \mathcal{S}_{i+1}(\mathcal{K}, n) \bmod a$, where $\mathcal{S}_i$ is the ith number in the sequence generated by cryptography pseudorandom sequence generator $\mathcal{S}$ seeded by $\mathcal{K}|n$. The outer code is pseudorandom and hidden from the buyer point of view, but deterministic to the merchant.

Fingerprint insertion is key-based as before. The fingerprint $\mathcal{F}$ is embedded (see Fig. 1) except that 1) three additional parameters are provided: the length of watermarking part L_1, maximum coalition size c, and maximum false detection rate ϵ in tracing a coalition; and 2) generation

of the fingerprint $\mathcal{F}$ of buyer n (i.e., line 1 in Fig. 1) is replaced by the concatenation of the watermark part $\mathcal{F}_1$ and the fingerprint part $\mathcal{F}_2$.

For fingerprint detection, we still use the algorithm shown in Fig. 2 except that subroutine *detect* in the algorithm is revised. From a recovered fingerprint template $\mathcal{F}$, its watermark part $\mathcal{F}_1$ is first checked against the codeword used in fingerprint insertion. If there is a single bit mismatch, the detection procedure returns none suspected. If the watermark part passes the first phase examination, the fingerprint part $\mathcal{F}_2$ will become the input of the tracing algorithm proposed in [3] for identifying a traitor.

5.3 Analysis

We have the following robustness results for our collusion resistant fingerprinting scheme. If the input of our detection algorithm is a pirated data copy under collusion attack only, the watermarking part will be detected correctly because the collusion attack can change the values only if the coalition has data copies that differ in those values. Therefore, the fingerprint part will become the input of the tracing algorithm proposed by Boneh and Shaw [3]. The tracing algorithm will return exactly one buyer; the probability that this returned buyer is a traitor in the coalition is greater than $1-\epsilon$, as proven in [3]. Therefore, the false miss fm, misattribution false hit fh^A, and false negative fn satisfy

$$fm = fh^A \leq \epsilon, fn = 0.$$

Now, consider the misdiagnosis false hit fh^D when the detection algorithm is applied to unmarked data. Note that the watermark part is the same for all buyers and it is examined first in the detection process. The probability of detecting a binary string for the watermark part is $\Pi_{i=0}^{L_1-1} 2B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5)$. Now, there is only one valid watermark codeword. Thus, the probability that the detected binary string matches the watermark codeword is $\frac{1}{2^{L_1}}$. Note that, whenever watermark detection succeeds, fingerprint detection returns exactly one valid buyer's id. Therefore,

$$\begin{aligned} fh^D &= \frac{1}{2^{L_1}} \Pi_{i=0}^{L_1-1} 2B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5) \\ &= \Pi_{i=0}^{L_1-1} B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5) \leq \frac{1}{2^{L_1}}. \end{aligned}$$

The misdiagnosis false hit has upper bound $\frac{1}{2^{L_1}}$ since $B(\lfloor \tau\omega_i \rfloor; \omega_i, 0.5) \leq 0.5$ when $\tau \in [0.5, 1)$. The upper bound is tight in the case that all ω_i are odd and $\tau = 0.5$. The misdiagnosis false hit rate can be reduced exponentially by increasing L_1 ($fh^D \simeq 0$ if $L_1 \gg 1$); it can also be decreased by increasing τ.

The length of the collusion resistant fingerprint is quite long even for small c and moderate ϵ. For example, for $c = 2$ (where at most two buyers can formulate a coalition), $\epsilon = 0.01$ and $N = 1,000$, the length of the collusion resistant codeword is 51,695; for $c = 3$ and 4, the length increases to 317,185 and 1,098,622, respectively. Such long fingerprints[2] are only suitable for very large databases such as terabyte scientific databases, surveillance databases, and antiterror databases [20]. If relatively short fingerprints[3] are used in fingerprinting (with relatively small c and moderate ϵ), one may ask what happens when more than c buyers collude. Experimental results for this question are reported in the next section.

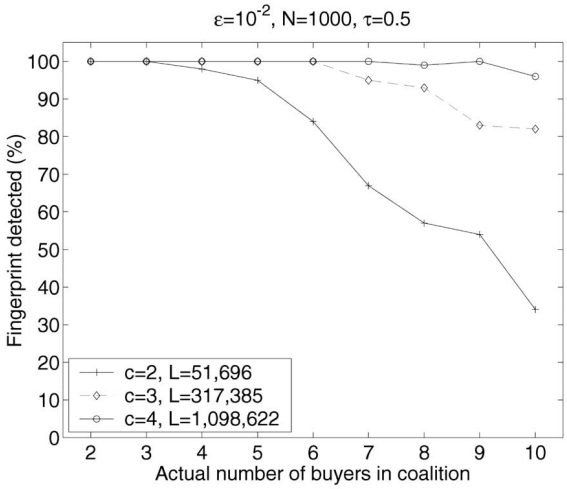

Fig. 3. Fingerprint detection under collusion attack.

5.4 Experiments

We tested our collusion resistant scheme with pirated data copies (thus, we set $L = L_2$ in testing) generated by coalitions whose sizes may be larger than c. During collusion attacks, the traitors in a coalition employ the "majority" strategy [3], that is, the traitors compare their copies and replace each bit position where the copies differ with the value shared by a majority of the copies (a random choice is made if zero and one appear in the same number of copies).

We tested our scheme 100 times for each case with different, randomly selected groups of buyers in collusion attacks. In order to control the test environment, we use synthetic data. The synthetic data consists of a primary key attribute as well as a single numerical attribute (i.e., $\nu = 1$). The primary key attribute is simply the sequence number of each tuple starting from 1, while the numerical attribute is a random variable uniformly distributed in the range of $[0, 1]$. Each buyer's fingerprint is embedded into the last bits of selected values (i.e., $\xi = 1$). The other parameters that are used in our experiments are: $\gamma = 5$, $\eta/(\gamma L) = 50$, $N = 1,000$, and $\tau = 0.5$.

The experimental results for collusion attacks are shown in Figs. 3 and 4. These figures illustrate that the detection rate does not drop dramatically as the coalition size goes beyond c. A high detection rate can be obtained by either increasing c or decreasing ϵ. Except for the power of BoSh code, our embedding scheme mitigates the damage of

2. All collusion resistant fingerprints are intrinsically long and there is no significant improvement to BoSh codes so far.

3. A possible solution to reduce the length of the fingerprint is to group database buyers, assuming that only buyers in each group will collude while members from different groups will not collude (e.g., buyers from competitor companies may not collude). However, because the length of BoSh codes is loglinear in N, reducing N is not very effective. For example, if $N = 1,000$ buyers are partitioned into 50 groups with 20 buyers per group, the length of BoSh code changes from 51,696 to 34,353 (still long) for $c = 2$ and $\epsilon = 0.01$.

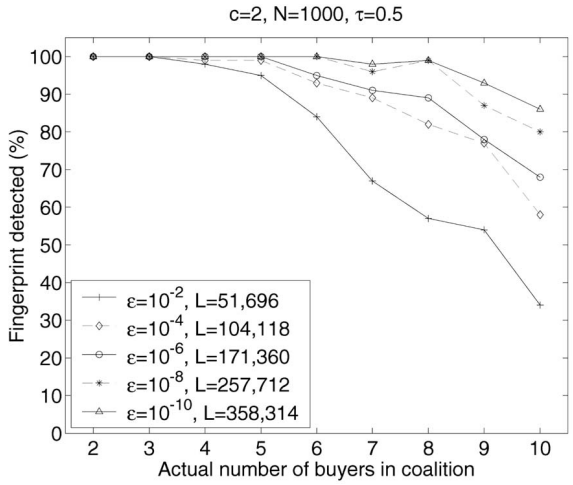

Fig. 4. Fingerprint detection under collusion attack.

collusion attacks by embedding multiple copies of each fingerprint bit and recovering it with a majority vote.

6 Extensions

There are several extensions that can be added to our fingerprinting scheme. In this section, we discuss fingerprinting codes that resist errors, fingerprinting without a primary key, and other methods for embedding a bitstring to a database relation.

6.1 Error-Resistant Fingerprinting Codes

Our scheme is not restricted to a particular form of fingerprint codes. Other collusion resistant codes can be easily incorporated into our scheme as long as their tracing algorithms detect collusion attacks with high probabilities. Recall that the effectiveness of BoSh code relies on the Marking Assumption, which states that colluding traitors cannot change undetected marks of fingerprints. In [10], Guth and Pfitzmann have extended this assumption to the case that all embedded marks, including undetected ones, are erasable. The fingerprinting code they designed is both error and collusion secure.

If we apply such a code in a similar way as we apply BoSh code, the code itself will have the power to detect not only the collusion attack, but also the bit-flipping attack. Note that the bit-flipping attack changes some embedded marks that cannot be detected by colluding buyers. This gives extra power to our detection algorithm, which employs a majority vote for resisting bit-flipping attack.

Similarly, traitor tracing has been investigated for shortened and corrupted fingerprints [21]. This complements our work on fighting against subset attack and bit-flipping attack. It is thus meaningful to incorporate these fingerprinting codes into our scheme.

6.2 Fingerprinting without a Primary Key

Both the AK scheme and our scheme depend critically on primary key attributes. Hence, these techniques cannot embed marks in a database relation without a primary key attribute. In [13], we proposed constructing a virtual primary key from the most significant bits of some of each tuple's attributes. The actual attributes that are used to construct the virtual primary key differ from tuple to tuple, and the attribute selection is based on the secret key only. To be consistent with the key-based property, the selection does not depend on an a priori ordering over the attributes, or on knowledge of the original relation or fingerprint codeword.

In particular, the bits of every numerical value $r.A_i$ in tuple r are partitioned into two parts: 1) $mb(r.A_i)$: the ξ least significant bits within which a fingerprint bit may be embedded, 2) $vpk(r.A_i)$: the remaining bits which are used to construct virtual primary key. The virtual primary key $r.V$ is constructed by concatenating the two (or more) hash values in $\{|\mathcal{H}(\mathcal{K}|vpk(r.A_i)| : i = 0, \ldots, \nu - 1\}$ that are closest to zero.

Since the attribute selection is dynamic, it is difficult for an attacker to destroy the virtual primary key through value modification or attribute deletion. However, unlike a primary key, the virtual primary key may not be unique for each tuple; consequently, the fingerprint bits may not be embedded evenly into the underlying data, and this renders the scheme weaker against attacks. The reader is referred to [13] for more details.

6.3 Other Embedding Methods

There have been other methods on how to embed a bitstring to a database relation. In [16], a database relation is transformed to a two-dimensional image which is divided into many small blocks. Each block is used to embed a single fingerprint bit. Within a block, different buyer's fingerprint bits are embedded into different positions. In another work [27], a bitstring is embedded by adding or not adding a set of extra tuples called stealth tuples. The stealth tuples are generated pseudorandomly and independently, according to the distribution of the original database relation.

While these methods are interesting, several precautions have to be taken (as mentioned in this paper) when they are used for collusion resistant fingerprinting. The first method is not collusion secure as it violates the Marking Assumption: Not only can colluding buyers detect fingerprint bits that have different values, but, also, many of those have the same values as they are embedded in different positions. Both methods need to be incorporated with a combination of multibit watermarking code and collusion secure fingerprinting code so as to avoid large false misdiagnosis false hit. In addition, both schemes can adopt our key-based approach in adapting BoSh code so that there is no need to remember all fingerprint codes or the random permutation in BoSh code (both are expensive due to the length of the code).

7 Related Work

Most of the research on watermarking and fingerprinting has targeted multimedia data (e.g., [5], [6], [11], [12]). However, the techniques developed for multimedia data cannot be extended to relational data since the data

TABLE 2
Change in Variance Introduced by Fingerprinting

		γ	100		50		25		12	
		ξ	4	8	4	8	4	8	4	8
Attribute	**Mean**	**Variance**								
Elevation	2959	78391		+1		+2		+5		+13
Aspect	156	12525		+2		+3		+6		+15
Slope	14	56								
HD-Hydrology	269	45177		+2		+3		+6	+1	+12
VD-Hydrology	46	3398						+1		+1
HD-Roadways	2350	2431272		+5		+1	+1	+19	-1	+17
Hillshade-9am	212	717		+1		+2		+4		+9
Hillshade-noon	223	391		+1		+3		+5		+11
Hillshade-3pm	143	1465		+1		+2		+5		+9
HD-Fire-Points	1980	1753490	-1	-6		-9	-1	-6		+28

After rounding to the nearest integer, the mean does not change except that it decreases by one for attribute Hillshade-3pm when $\gamma = 12$ and $\xi = 8$.

properties, as well as data operations, differ significantly in these two areas. For example, multimedia data components are highly correlated, while database tuples can be manipulated independently. Multimedia techniques are designed to be robust against operations such as zooming and compression, while a fingerprinting scheme should be robust against typical database operations such as tuple addition, deletion, and modification.

Watermarking and fingerprinting techniques have also been proposed for other kinds of data such as computer programs (e.g., [8]) and text (e.g., [4], [28]). However, those techniques are based on either hiding a mark in the visual representation of a program or text or transforming a program or text into a semantically equivalent alternate representation. Those techniques are not appropriate for marking database relations.

Perhaps the most closely related work is the watermarking of relational databases [1], [9], [14], [23], [24]. Due to the similarity between watermarking and fingerprinting, most of the techniques developed in this area are complementary to our work. As an initial effort, we extended the watermarking scheme [1] to fingerprinting relational data. We identified the specialties in fingerprinting (e.g., robustness measures and collusion attacks) and evaluated our scheme quantitatively.

Fingerprinting has been studied extensively in cryptography literature, with a focus on how to design collusion resistent fingerprint codes (e.g., [3], [10], [15], [17], [18], [19], [26]). This line of work is also complementary to ours as it provides well-designed fingerprint codes (i.e., the fingerprint part in our scheme) and we present how to embed (extract) such codes into (from) relational databases. As indicated in Section 5, however, careful adaptation is required as a straightforward incorporation may cause severe problems such as 100 percent misdiagnosis false hit.

8 Conclusion

In this paper, we presented a fingerprinting technique for relational databases by extending the AK scheme in a significant way. Our contribution is unique in the following two aspects:

- We extend the AK scheme to embedding an arbitrary bitstring as fingerprint. The extended scheme shares the same set of assumptions and properties. Different measures, such as misattribution false hit and misdiagnosis false hit, are identified for evaluating a fingerprinting scheme. A quantitative analysis is presented for all the evaluation measures under representative types of attacks.
- We show that simply incorporating collusion resistant code from cryptography literature does not work as it may produce a 100 percent misdiagnosis false hit when applied to unmarked data. We solve the problem by combining multibit watermark with collusion resistant code in our scheme.

We mention that our scheme is symmetric since both a merchant and a buyer possess the same marked data copy. In asymmetric fingerprinting schemes (e.g., [18], [19]), only the buyer knows the marked data copy, so a merchant can use the mark detection process to prove to a third party that the data was pirated and that the pirated data was sold to a specific buyer. Asymmetric fingerprinting schemes may be constructed based on symmetric fingerprinting schemes and public key cryptographic primitives in a two-party protocol.

Appendix

Experimental Results on Errors

We present experimental results on the errors introduced by fingerprint insertion. The experiments were performed on a real-life data set, the Forest Cover Type data set, available at http://kdd.ics.uci.edu/databases/covertype/covertype. html. The data set has always been used in [1] for evaluating the AK scheme. The data set has 581,012 tuples, each with 61 attributes and no primary key. The first 10 integer-valued attributes are chosen for embedding

fingerprints (i.e., $\nu = 10$). We added an extra attribute, called *id*, to serve as the primary key.

Table 2 illustrates the impact of fingerprint insertion on the mean and variance of the values of marked attributes. The change in these statistics validates our assertion that the errors introduced by fingerprinting are minuscule.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments.

REFERENCES

[1] R. Agrawal, P.J. Haas, and J. Kiernan, "Watermarking Relational Data: Framework, Algorithms and Analysis," *The VLDB J.*, vol. 12, no. 2, pp. 157-169, 2003.
[2] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Comm. ACM*, vol. 13, no. 7, pp. 422-426, 1970.
[3] D. Boneh and J. Shaw, "Collusion Secure Fingerprinting for Digital Data," *IEEE Trans. Information Theory*, vol. 44, no. 5, pp. 1897-1905, 1998.
[4] J. Brassil, S. Low, N. Maxemchuk, and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying," *Proc. Conf. Computer Comm., 13th Ann. Joint Conf. IEEE Computer and Comm. Socs., Networking for Global Comm. (INFOCOM)*, pp. 1278-1287, 1994.
[5] I. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Trans. Image Processing*, vol. 6, no. 12, pp. 1673-1687, 1997.
[6] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*. Morgan Kaufmann, 2001.
[7] S. Crawer, N. Memon, B. Yeo, and M. Yeung, "Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitations, Attacks, and Implications," *IEEE J. Selected Areas in Comm.*, vol. 16, no. 4, pp. 573-586, 1998.
[8] D. Glover, *The Protection Of Computer Software*, second ed. Cambridge Univ. 1992.
[9] D. Gross-Amblard, "Query-Preserving Watermarking of Relational Databases and XML Documents," *Proc. ACM Symp. Principles of Database Systems (PODS)*, pp. 191-201, 2003.
[10] H.-J. Guth and B. Pfitzmann, "Error- and Collusion-Secure Fingerprinting for Digital Data," *Proc. Conf. Information Hiding (IH '99)*, pp. 134-145, 2000.
[11] N. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking-Attacks and Countermeasures*. Kluwer, 2000.
[12] *Information Hiding Techniques for Steganography and Digital Watermarking*, S. Katzenbeisser and F. Petitcolas, eds. Artech House, 2000.
[13] Y. Li, V. Swarup, and S. Jajodia, "Constructing a Virtual Primary Key for Fingerprinting Relational Data," *Proc. ACM Workshop Digital Rights Management (DRM)*, pp. 133-141, 2003.
[14] Y. Li, V. Swarup, and S. Jajodia, "Robust Watermarking Schemes for Relational Data," submitted for publication.
[15] T. Lindkvist, "Fingerprinting Digital Documents," Linköping Studies in Science and Technology, thesis no. 798, 1999.
[16] S. Liu, S. Wang, R. Deng, and W. Shao, "A Block Oriented Fingerprinting Scheme in Relational Database," *Proc. Seventh Ann. Int'l Conf. Information Security and Cryptology (ICISC)*, 2004.
[17] B. Pfitzmann and A.-R. Sadeghi, "Coin-Based Anonymous Fingerprinting," *Advances in Cryptology—EUROCRYPT 1999, Int'l Conf. the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 150-164, 1999.
[18] B. Pfitzmann and M. Schunter, "Asymmetric Fingerprinting (Extended Abstract)," *Proc. Advances in Cryptology—EUROCRYPT 1996, Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 84-95, 1996.
[19] B. Pfitzmann and M. Waidner, "Asymmetric Fingerprinting for Larger Collusions," *Proc. ACM Conf. Computer and Comm. Security (CCS)*, pp. 151-160, 1997.
[20] W. Roush, "Managing Antiterror Databases," *Technology Rev.*, p. 22, June 2003.
[21] R. Safavi-Naini and Y. Wang, "Traitor Tracing for Shortened and Corrupted Fingerprints," *Proc. ACM Workshop Digital Rights Management (DRM)*, pp. 81-100, 2002.
[22] B. Schneier, *Applied Cryptography*, second ed. John Wiley and Sons 1996.
[23] R. Sion, "Proving Ownership Over Categorical Data," *Proc. IEEE Int'l Conf. Data Eng. (ICDE)*, pp. 584-596, 2004.
[24] R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD)*, pp. 98-108, 2003.
[25] L. Vaas, "Putting a Stop to Database Piracy," *eWEEK, Enterprise News and Revs.*, Sept. 2003, http://www.eweek.com/print_article/0,3048,a=107965,00.asp.
[26] Y. Yacobi, "Improved Boneh-Shaw Content Fingerprinting," *Proc. Topics in Cryptology—CT-RSA 2001, the Cryptographer's Track at RSA Conf. (CT-RSA)*, pp. 378-391, 2001.
[27] K. Yoshioka, J. Shikata, and T. Matsumoto, "A Method of Database Fingerprinting," *Proc. 2004 Workshop Information Security Research*, 2004.
[28] B. Zhu, J. Wu, and M. Kankanhalli, "Print Signatures for Document Authentication," *Proc. ACM Conf. Computer and Comm. Security (CCS)*, pp. 145-154, 2003.

Yingjiu Li received the PhD degree in Information Technology from George Mason University in 2003. He is currently an assistant professor in the School of Information Systems at Singapore Management University. His research interests include data security, privacy, and digital rights management. He is a member of the ACM and the IEEE.

Vipin Swarup received the BTech degree from the Indian Institute of Technology, Mumbai, and the MS and PhD degrees from the University of Illinois at Urbana-Champaign. He is a principal scientist in the Security and Information Operations Division at The MITRE Corporation. He has been the principal investigator of research projects in trust management, mobile agent security, security guards, and type theory. His current research interests include secure information sharing, detecting insider threat behavior, credential management, and vulnerability analysis.

Sushil Jajodia is the BDM International Professor of Information Technology and the director of Center for Secure Information Systems at George Mason University, Fairfax, Virginia. His research interests include information security, temporal databases, and replicated databases. He has authored five books, edited 22 books, and published more than 250 technical papers in the refereed journals and conference proceedings. The URL for his Web page is csis.gmu.edu/faculty/jajodia.html. He is a senior member of the IEEE.