Constructing a Virtual Primary Key for Fingerprinting Relational Data

Yingjiu Li[*]
George Mason University
4400 University Drive
Fairfax, VA 22030
yjli@smu.edu.sg

Vipin Swarup
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
swarup@mitre.org

Sushil Jajodia
George Mason University
4400 University Drive
Fairfax, VA 22030
jajodia@gmu.edu

ABSTRACT

Agrawal and Kiernan's watermarking technique for database relations [1] and Li et al's fingerprinting extension [6] both depend critically on primary key attributes. Hence, those techniques cannot embed marks in database relations without primary key attributes. Further, the techniques are vulnerable to simple attacks that alter or delete the primary key attribute.

This paper proposes a new fingerprinting scheme that does not depend on a primary key attribute. The scheme constructs virtual primary keys 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. Attribute selection is based on a secret key that is known to the merchant only. Further, the selection does not depend on an apriori ordering over the attributes, or on knowledge of the original relation or fingerprint codeword.

The virtual primary keys are then used in fingerprinting as in previous work [6]. Rigorous analysis shows that, with high probability, only embedded fingerprints can be detected and embedded fingerprints cannot be modified or erased by a variety of attacks. Attacks include adding, deleting, shuffling, or modifying tuples or attributes (including a primary key attribute if one exists), guessing secret keys, and colluding with other recipients of a relation.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*security, integrity, and protection*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Algorithms, Performance

Keywords

Fingerprinting, relational databases, primary keys, robustness

[*]Yingjiu Li's current address: School of Information Systems, Singapore Management University, 469 Bukit Timah Road, Singapore 259756

1. INTRODUCTION

Fingerprinting is a class of information hiding techniques that can help protect data from threats such as unauthorized disclosure. Consider a generic scenario where merchants sell digital data to buyers. Some dishonest buyers (called traitors) may redistribute the data to others (called pirates) without permission from the merchants. A merchant 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.

Fingerprinting is often discussed in comparison or extension to watermarking. Watermarking is another class of information hiding techniques whose purpose is to identify the sources of data. A merchant may use a watermarking scheme to embed a merchant-specific mark into her data and assert ownership of the data by detecting the watermark. Thus, watermarking is used to embed marks that identify the merchant while fingerprinting is used to embed marks that identify legitimate buyers.

Watermarking and fingerprinting have been studied extensively in the context of multimedia data (still images, audio, and video). Recently, algorithms have been proposed for watermarking database relations [1, 8, 4]. Agrawal and Kiernan [1] present a novel technique for embedding watermarks within numeric attributes of relations. They assume 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 values with five decimal places, then a buyer may accept that 0.1% of the tuples have arbitrary fractional errors as long as the rest are accurate up to five decimal places.

Li et al. [6] have generalized Agrawal and Kiernan's watermarking technique to enable the fingerprinting of relational data. The fingerprinting technique enables a buyer-specific bitstring to be embedded and extracted from a database relation, as compared to the watermarking technique which enables a single watermark bit to be embedded and extracted from a relation. The schemes are robust against various attacks including flipping bits, adding or deleting tuples, guessing secret keys, and colluding with other recipients.

Both Agrawal and Kiernan's watermarking technique and Li et al.'s fingerprinting technique rely on a critical assumption, namely that a database relation has a primary key attribute that either does not change or else can be recovered. The rationale behind this assumption is that a primary key attribute contains essential information and that modification or deletion of this information will substantially reduce the data's value. However, as a consequence of this assumption, those techniques cannot embed marks in database relations without primary key attributes. Further, the techniques are vulnerable to attacks that alter or delete the primary key attribute.

The main contributions of this paper are to propose new schemes for fingerprinting relational data without depending on a primary key attribute, and to demonstrate analytically that the schemes are robust against a wide range of attacks including attacks that alter, shuffle, or delete any attributes. We propose two schemes (E-scheme and M-scheme) that construct virtual primary keys from the most significant bits of some of each tuple's attributes; the virtual primary keys are then used in fingerprinting as in previous work [6].

In M-scheme, the actual attributes that are used to construct the virtual primary key differ from tuple to tuple. Attribute selection is based on a secret key that is known to the merchant only. Further, the selection does not depend on an apriori ordering over the attributes (unlike the previous schemes which do depend on a fixed ordering of attributes). Thus, M-scheme is robust against attacks that delete, shuffle, or modify attributes (including a primary key attribute if one exists), or that attempt to alter the virtual primary keys. Our analysis demonstrates that an attacker must modify or delete a large number of attributes in order to alter or delete the embedded mark.

The remainder of this paper is organized as follows. Section 2 reviews a scheme [6] for fingerprinting relational data with primary keys. Section 3 presents several schemes for fingerprinting relational data without relying on real primary keys. Section 4 presents a detailed analysis of the new fingerprinting schemes. We examine desirable properties such as imperceptibility and detectability. Section 5 describes several attack classes and analyzes the behavior of the extended fingerprinting schemes under these attacks. Section 6 discusses whether a database relation is fingerprintable and how to choose among different fingerprinting schemes. Section 7 presents experimental results from our prototype implementation. Section 8 summarizes the paper and suggests possible future directions.

2. FINGERPRINTING RELATIONS WITH PRIMARY KEYS

In this section, we briefly review a fingerprinting scheme for relational data [6] based on which we shall derive our new scheme. Since the basic data objects used in that scheme are tuples, we refer to it as *tuple-level scheme*, or simply *T-scheme*. T-scheme is based on Agrawal and Kiernan's watermarking scheme [1].

2.1 Notation and Parameters

Consider a database relation R that has a single primary key attribute P and ν numerical attributes $A_0, \ldots, A_{\nu-1}$. Without loss of generality, let the schema of R be $R(P, A_0, ..., A_{\nu-1})$ and let the database have η tuples. For each attribute value $r.A_i$ of tuple $r \in R$, one of its $\xi(r.A_i)$ least significant bits could be used to embed a fingerprint bit. $\xi(r.A_i)$ could depend on the number of bits in a standard binary representation of $r.A_i$, or it could be a constant number that is independent of the value $r.A_i$. To be simple, we use ξ for $\xi(r.A_i)$ unless otherwise stated.

Let N be the number of users (or buyers) to whom the data is being distributed. A fingerprint (codeword) $\Gamma = (f_0, \ldots, f_{L-1})$ is a binary string with length $L \geq \log N$. Each user is assigned a unique fingerprint with the same length L. The set of all fingerprints is called the code book. A fingerprint is embedded into each copy of R and the fingerprinted data is then distributed to the corresponding user.

User n's fingerprint is computed by a cryptographic hash function $\mathcal{H}_0$ whose input is the concatenation of a secret key $\mathcal{K}$ (known by the merchant only) and user identifier n. The output of $\mathcal{H}_0$ is a binary string of length L. We shall assume that this results in a unique fingerprint for each user $n = 0, \ldots, N-1$. This is usually the case when $L > \log N$ because of the collision-free property of the hash function. If collisions do exist, we may use a larger L, reserve the user identifiers that cause collision, or change the secret key.

We also use the following cryptographic hash functions in our fingerprinting schemes:

- First Hash: $\mathcal{H}_1(\mathcal{K}, r.P) = \mathcal{H}(\mathcal{K} \circ r.P)$
- Message Authenticated Code: $\mathcal{H}_2(\mathcal{K}, r.P) = \mathcal{H}(\mathcal{K} \circ \mathcal{H}(\mathcal{K} \circ r.P))$

where $\mathcal{K}$ is a secret key, $r.P$ is the value of the primary key attribute P of tuple r in relation R, $\mathcal{H}$ is a standard hash function (e.g., MD5 or SHA), and $\circ$ denotes concatenation.

2.2 Fingerprinting Scheme

Li et al [6] describe a scheme (which we call T-scheme in this paper) for fingerprinting relational data with primary key attributes. T-scheme assumes that changing a small number of attribute bits is imperceptible and does not degrade the value of the data. It also assumes that changing the primary key attribute or changing a large number of other attribute bits does degrade the value of the data substantially.

For fingerprint insertion, T-scheme scans each tuple and uses the message authenticated code $\mathcal{H}_2$ to select some attribute bits in some tuples for modification. It first selects one out of γ tuples: a tuple is selected if $\mathcal{H}_2(\mathcal{K}, r.P)$ mod γ equals 0. Then, for each selected tuple, it selects one out of ν attributes: attribute i is selected if $i = \mathcal{H}_2(\mathcal{K}, r.P)$ mod ν. For each selected attribute, it selects one out of ξ bits: the j-th least significant bit is selected if $j = \mathcal{H}_2(\mathcal{K}, r.P)$ mod ξ. Hash function $\mathcal{H}_2$'s properties ensure that the selections are made uniformly. Further, the selections depend on a secret key $\mathcal{K}$. The scheme then replaces the selected bit of the selected attribute in the selected tuple with a computed mark bit.

T-scheme computes the mark bit by selecting one out of the L fingerprint bits and masking the selected fingerprint bit with a uniformly distributed mask bit. It uses the cryptographic hash function $\mathcal{H}_1$ in the selection of the fingerprint bit: bit f_l in fingerprint Γ is selected if $l = \mathcal{H}_1(\mathcal{K}, r.P)$ mod L. It also uses $\mathcal{H}_1$ in computing the mask bit: the mask bit is zero if $\mathcal{H}_1(\mathcal{K}, r.P)$ is even; it is one otherwise. It XOR's the selected fingerprint bit with the mask bit to get the mark bit. Note that the scheme uses two different hash functions to compute the attribute bit location and the mark bit value; this ensures that the fingerprint bits are not correlated with the locations in which they are embedded. On average, each fingerprint bit is embedded into a database w times where $w = \eta/(\gamma L)$.

For fingerprint detection, T-scheme scans each tuple and computes the mask bit and the marked position in exactly the same way as in fingerprint insertion. Then it extracts the embedded fingerprint bit by XORing the mask bit with the bit at the marked position in the tuple. Recall that each fingerprint bit f_i is inserted multiple times; it should be detected the same number of times if the data has not changed. However, due to various possible attacks or benign updates, we expect that not all but a majority of the embedded values for f_i will be recovered. The scheme uses a majority vote in fingerprint detection: two counting variables $count[i][0]$ and $count[i][1]$ indicate the numbers of 0's and 1's respectively that have been recovered for a fingerprint bit f_i; after all tuples have been processed, fingerprint bit f_i is set to 0 if the ratio of recovered 0's (i.e., $count[i][0]/(count[i][0] + count[i][1])$) is greater than τ, and similarly for 1's. Parameter $\tau \in [0.5, 1]$ is the minimum fraction of correctly marked tuples needed for detection of each fingerprint bit.

A traitor is detected if the recovered fingerprint $\Gamma = (f_0, \ldots, f_{L-1})$ matches one of the N users' fingerprints which can be computed by the hash function $\mathcal{H}_0$ on the secret key and user identifiers. Clearly, if no embedded bit is recovered for some f_i (if neither the ratio of 0's nor the ratio of 1's is greater than τ), then no fingerprint is detected.

T-scheme is robust against attacks that update, delete or modify tuples due to the following reasons: (i) every tuple is marked independently; (ii) every fingerprint bit is embedded multiple times; and (iii) a majority vote is used in fingerprint detection. Li et al. [6] analyze the robustness and detectability properties of this scheme.

3. FINGERPRINTING RELATIONS WITHOUT PRIMARY KEYS

T-scheme uses a secret key in every step of fingerprint insertion. Thus, a user is unable to determine where the fingerprint is embedded, the embedded values, the relationship between the embedded values and corresponding fingerprint bits, and even the fingerprint codeword. However, T-scheme depends critically on a primary key attribute. That is, it assumes that a relation has a primary key attribute, and that modifications to the primary key cause substantial degradation in the value of the relation. In this section, we propose fingerprinting schemes that do not require these assumptions. These new schemes also address another minor deficiency of T-scheme: T-scheme depends on an ordering over the attributes in a relation while these new schemes do not.

3.1 S-Scheme: A Simple Solution

Agrawal and Kiernan [1] describe a simple extension that enables their watermarking technique to apply to relations without primary keys. Our fingerprinting technique T-scheme can be modified in a similar way to fingerprint relations without primary keys. Assume that a relation R consists of a single numerical attribute A. For each tuple r, partition the bits of $r.A$ into two parts: (i) $mb(r.A)$: the ξ bits of element $r.A$ within which a fingerprint bit may be embedded; and (ii) $vpk(r.A)$: the remaining bits of element $r.A$ which are used as a virtual primary key for the tuple r. T-scheme can then be used for fingerprinting by using the virtual primary key in place of a real primary key.

If the "virtual primary key" $vpk(r.A)$ is unique to every tuple r, on average η/γ bits are used for fingerprinting, which is the same as in T-scheme. Further, the hash values $\mathcal{H}_2(\mathcal{K},\, vpk(r.A))$ are uniformly distributed and thus each fingerprint bit is embedded on average $\eta/(\gamma L)$ times. However, $vpk(r.A)$ is not a true primary key and may not be unique for each tuple r. With duplicate virtual primary keys, the average number of modified bits is no longer η/γ and each fingerprint bit is no longer embedded into data an average of $w = \eta/(\gamma L)$ times. In fact, some fingerprint bits may be embedded fewer times than others, rendering the scheme susceptible to attacks such as bit-flipping attacks (see Section 5.2). Some fingerprint bits may not be embedded at all, in which case fingerprint detection will always fail. We call the problem of duplicate virtual primary keys the *duplicate problem*.

Another issue is that since the attribute A is not a real primary key, an attacker may be able to drop or modify the attribute (and hence foil detection of an embedded fingerprint) without significantly degrading the data. We call this the *deletion problem*.

If a relation has n numerical attributes, then we can select any one attribute to generate the virtual primary key as above and use the remaining attributes for marking. The duplicate problem can be mitigated by selecting the attribute with the fewest duplicates to provide the virtual primary key. More generally, the virtual primary key can be constructed by combining bits from several attributes so as to minimize duplicates. However, these approaches do not solve the deletion problem—an attacker can still defeat the scheme by deleting or modifying any of the bits which are used to construct the virtual primary key.

3.2 E-Scheme: Element-Based Fingerprinting

S-scheme processes each tuple in a relation and decides whether or not to embed a fingerprint bit in the tuple. In contrast, E-scheme examines each numerical attribute in each tuple independently, computing a virtual primary key from the attribute value, then using the computed key to decide whether or not to embed a fingerprint bit in the attribute value.

For each element $r.A_i$ in each tuple r, E-scheme uses the virtual primary key $vpk(r.A_i)$ to determine whether, where, and how to embed or extract a fingerprint bit in the marking part $mb(r.A_i)$ of the same element. Recall that T-scheme selects on average one bit per γ tuples, where each tuple has ν attributes (i.e., elements) for fingerprinting. E-scheme maintains the same ratio by selecting one bit per $\gamma\nu$ elements. Thus, E-scheme selects an element $r.A_i$ for marking if $\mathcal{H}_2(\mathcal{K}, vpk(r.A_i)) \bmod \gamma\nu$ equals 0; if $r.A_i$ is selected, E-scheme selects the j-th least significant bit in $mb(r.A_i)$ where $j = \mathcal{H}_2(\mathcal{K}, vpk(r.A_i)) \bmod \xi$. Finally, E-scheme embeds (or extracts) a mark in the selected bit as in T-scheme. It is clear that E-scheme does not rely on a real primary key attribute or on the order of attributes.

E-scheme mitigates the deletion problem since deletion of any subset of attributes will not erase an embedded fingerprint or prevent its detection. However, this scheme is still vulnerable to the duplicate problem since the numerical attributes are processed independently despite not being real primary keys.

3.3 M-Scheme: A Robust Scheme

Like S-scheme, M-scheme marks each tuple based on the tuple's virtual primary key. However, while S-scheme (and E-scheme) uses the same bit positions in all tuples to construct virtual primary keys, M-scheme dynamically selects the bit positions used to construct a virtual primary key. This makes M-scheme robust against attacks that attempt to modify or delete only the bits used to construct the virtual primary key.

As in S-scheme, the bits of every numerical element $r.A_i$ in tuple r are partitioned into two parts: $vpk(r.A_i)$ and $mb(r.A_i)$. For each tuple $r = (r.A_0, \ldots, r.A_{\nu-1})$, M-scheme computes the virtual primary key $r.V$ by concatenating the two hash values in $\{|\mathcal{H}_1(\mathcal{K}, vpk(r.A_i))| : i = 0, \ldots, \nu - 1\}$ that are closest to zero (assume $\nu \geq 2$). In general, we may concatenate more than two hash values to optimize the construction of the virtual primary key.

M-scheme then uses the T-scheme technique to process the tuple, but with two modifications. First, M-scheme uses the virtual primary key instead of a real primary key. Second, for each selected tuple in fingerprint insertion or detection, attribute A_i is selected for embedding if its hash value $|\mathcal{H}_1(\mathcal{K},\, vpk(r.A_i))|$ is closest to zero among all attributes' hash values. Multiple attributes may be selected in a tuple if they map to the same lowest hash value. In comparison, T-scheme selects attribute A_i if $i = \mathcal{H}_2(\mathcal{K}, r.P) \bmod \nu$. Thus, while T-scheme depends on the primary key and on the order of attributes, M-scheme does not.

The construction of the virtual primary key is dynamic and content based. It is dynamic because for different tuples, different attributes are selected to form the virtual primary key. Without knowing the secret key, an attacker is unable to determine which attributes are selected in each tuple. The construction is also content-based; it depends on the hash values rather than the order of the attributes. Because of these two properties, an attacker is unable to destroy enough virtual primary keys through attribute modification unless he modifies a large number of attributes (see Section 5.2).

4. ANALYSIS

A fingerprinting (or watermarking) scheme for relational data must satisfy several properties in order to be useful. For instance, it should enable insertion and detection of fingerprints, it should be robust against a broad class of attacks, and it should ensure that innocent users are not identified as traitors (whether inadvertently or due to attacks). Such properties have been identified in [1] and [6]. In this section, we summarize those properties and focus on the new issues introduced by E-scheme and M-scheme.

4.1 Duplicate Index

In T-scheme, if the hash values of primary key attributes are perfectly uniform, each fingerprint bit f_l is embedded into data exactly $w = \eta/\gamma$ times. However, due to the duplicate problem especially in E-scheme and M-scheme, the fingerprint bits are not embedded evenly: some may be embedded more often and some may not be embedded at all. Let w_l be the actual times that each fingerprint bit f_l is embedded ($l = 0, \ldots, L-1$). Let $w_{max} = \max_l w_l$ and $w_{min} = \min_l w_l$. We use *duplicate index* δ to measure the duplicate problem.

- Duplicate index: $\delta = (w_{max} - w_{min})/w_{min}$

If there is no duplicate problem (i.e., $w = w_0 = \ldots = w_{L-1} = \eta/(\gamma L)$), the duplicate index equals zero. This is the "ideal" case of T-scheme. If some fingerprint bit is not embedded into data due to the duplicate problem (i.e., $\min_l w_l = 0$), the duplicate index is infinity. The smaller the duplicate index, the more evenly the fingerprint is embedded.

While previous research on fingerprinting has focused on the case where the duplicate index equals zero [6], we study the case where the duplicate index is greater than zero. This is the case for E-scheme and M-scheme, and also the real case (not "ideal" case) for T-scheme. While the "ideal" case of T-scheme has been analyzed in [6], we analyze E-scheme, M-scheme, and the real case of T-scheme in the rest of this paper.

4.2 Imperceptibility

A critical property of any fingerprinting scheme is that embedded fingerprints should be imperceptible, i.e., the original data and fingerprinted data should be indistinguishable. As a consequence, embedding a fingerprint will not degrade the value of data. For multimedia data, imperceptibility requires that the human perceptual system should not be able to distinguish between marked and unmarked data [9]. Since database relations are not intended for direct human perception, we consider a different notion of imperceptibility that is based on the accuracy of data; that is, the extent to which a database relation is similar to a merchant's original unmarked relation. This includes errors in individual values, aggregate statistics (e.g., means and deviations) of attributes, correlations among attributes, and constraints across relations (e.g., foreign key constraints).

We have shown in prior work [6] that T-scheme embeds fingerprints in database relations by introducing a very small number of errors into the relations. The scheme is applicable to data where small changes in accuracy are not perceivable but large changes in accuracy are. The analysis given in [6] is based on that only η/γ attribute values are used for embedding a fingerprint. In comparison, the number of attribute values that are used for embedding fingerprint in E-scheme or T-scheme is given by η/γ' where $\gamma' = \frac{\eta}{\sum_l w_l}$ (w_l is the times that each fingerprint bit f_l is embedded). The same analysis on imperceptibility for T-scheme can be applied to E-scheme and M-scheme by simply replacing γ with γ'. The reader is referred to [6] for more details.

4.3 Incremental Updatability

In M-scheme, each tuple is marked independently of all other tuples. Hence, the addition, deletion, and modification of tuples can be handled efficiently by only recomputing and embedding fingerprints for new or updated tuples. In E-scheme, each element is marked independently and updatability is at the element level.

4.4 Key-Based System

According with Kerckhoffs' principle in security engineering, the entire design of the fingerprinting system may be made public, with the exception of secret keys. In our schemes, a secret key determines the fingerprint codebook, the marks to embed, the relationship between the fingerprint codewords and the embedded marks, and the positions at which the marks are embedded. Thus, the fingerprinting schemes remain robust even if the algorithms and other parameters are made public.

4.5 Blind System

A blind system means that a fingerprint detection algorithm does not require knowledge of the original database, embedded fingerprint, or codebook. Our fingerprinting schemes (E-scheme and M-scheme) are pure blind systems. Previous watermarking and fingerprinting schemes may require knowledge of the original database if the primary key or the order of the attributes are changed in an attack.

4.6 Detectability

In the absence of malicious attacks or benign updates, a fingerprinting scheme should be able to determine if data has been marked with a fingerprint and, if so, extract the correct embedded fingerprint. Further, it should not extract a fingerprint (purely by chance) from unmarked data.

It is clear that a correct fingerprint can be detected from unmodified marked data if and only if the duplicate index is not infinity (i.e., each fingerprint bit is embedded at least once). In the remainder of this subsection, we investigate the probability that a valid fingerprint is detected from unmarked data. We call this probability *misdiagnosis false hit*.

Recall that each codeword bit f_l is embedded in data w_l times. The bit f_l is detected as 0 if more than τw_l of its embedded copies are detected to be 0. If the unmarked data has uniformly distributed values at each of the fingerprintable bit positions, then each of the w_l bits is 0 with independent probability $p = 0.5$ (the case of $p \neq 0.5$ can be analyzed similarly since the imperceptibility analysis shows that fingerprinting causes minuscule changes to the data statistics).

We model this as Bernoulli trials (repeated independent trials) with probability p of success and q of failure. For $0 \leq k \leq n$, let $b(k; n, p) = \binom{n}{k} p^k q^{n-k}$ be the probability that n Bernoulli trials result in k successes and $n-k$ failures where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, and let $B(k; n, p) = \sum_{i=k+1}^{n} b(i; n, p)$ be the probability of having more than k successes in n Bernoulli trials. The probability of recovering a 0, i.e., the probability that more than τw_l of the w_l embedded bits (for codeword bit f_l) are 0, is $B(\lfloor \tau w_l \rfloor; w_l, 0.5)$. Since the probability of recovering a 1 is given similarly, and since each codeword bit is embedded independently, a codeword (i.e., a binary string) is detected in unmarked data with probability

$$\Pi_l(2 * B(\lfloor \tau w_l \rfloor; w_l, 0.5)) = 2^L \Pi_l B(\lfloor \tau w_l \rfloor; w_l, 0.5)$$

Note that N valid fingerprints in the codebook are selected pseudo-randomly by a hash function from 2^L possible codewords. Thus, the probability that the codeword extracted from unmarked data is a valid fingerprint is $\frac{N}{2^L}$, and the overall misdiagnosis false hit rate is:

$$\frac{N}{2^L} * 2^L \Pi_l B(\lfloor \tau w_l \rfloor; w_l, 0.5) = N * \Pi_l B(\lfloor \tau w_l \rfloor; w_l, 0.5)$$

If the duplicate index is not infinity (i.e., each fingerprint bit is embedded at least once), the misdiagnosis false hit rate must be less than or equal to $\frac{N}{2^L}$ (since $\tau \geq 0.5$ and hence $B(\lfloor \tau w_l \rfloor; w_l, 0.5) \leq 0.5$). This can be lowered exponentially by increasing the length L of the fingerprint.

5. ROBUSTNESS

A fingerprinting scheme should be robust against benign database operations and malicious attacks that may destroy or modify embedded fingerprints. That is, it should be hard for attacks and benign updates to erase embedded fingerprints; to modify embedded fingerprints so that an innocent user is implicated as a traitor; or to modify unmarked data so that the fingerprinting scheme detects a valid fingerprint codeword in the modified data. Benign operations include adding tuples, deleting tuples, and updating tuples in database relations. Malicious attacks include selective modifications of fingerprinted relations, and taking subsets of the relations, with the express purpose of modifying or erasing the embedded fingerprint.

We investigate *false miss rate*: the probability of failing to detect an embedded codeword correctly. The false miss rate is the sum of *false negative* and *misattribution false hit*. The false negative is the probability of detecting no valid codeword from fingerprinted data. And the misattribution false hit is the probability of detecting an incorrect but valid codeword from fingerprinted data.

Embedded marks can always be destroyed by making substantial modifications to marked data. Thus, we consider the robustness of our fingerprinting schemes relative to the data modifications made by attacks and updates. We assume that when attacks modify data, they also degrade the value of the data. Benign updates, however, usually enhance the value of data.

5.1 Attacks

Fingerprinting schemes are subject to several common attacks which have been identified in [1, 6]. Some attacks reduce the accuracy of data, e.g., by reducing the precision of attribute values or by introducing errors into the data. This type of attacks includes randomization attacks (some bits are assigned random values), zero out attacks (the values of some bits are changed to zero), bit flipping attacks (the values of some bits are inverted), rounding attacks (some bits are deleted due to the rounding of numerical values), and primary key attacks (primary key attribute values are deleted or modified).

Some attacks modify relations without reducing accuracy. This type of attacks includes subset attacks (only a subset of tuples or attributes of a fingerprinted relation appear in a pirated database) and superset attacks (new tuples or attributes are added to a fingerprinted relation).

Some attacks seek to provide a traitor or pirate with evidence that raises doubts about a merchant's claims. This type of attacks includes additive attacks (add an additional fingerprint to a pirated copy thus confusing a third party) and invertibility attacks (discover a fictitious fingerprint in a relation, e.g., by trying different secret keys to find one that matches the data by random chance).

Finally, some attacks, called *collusion attacks*, require attackers to have access to multiple fingerprinted copies of the same relation but with different embedded fingerprints. Collusion attacks include majority attacks (create a new relation with the same schema as the copies but with each bit value computed as the majority function of the corresponding bit values in all copies) and mix and match attacks (create a pirated copy by combining subsets of tuples and attributes from each fingerprinted copy).

If a fingerprint detection algorithm is applied to data which has been subject to attacks and benign updates, then the algorithm may possibly return no valid fingerprint (as measured by the false negative rate) or it may possibly return a valid but incorrect fingerprint (as measured by the misattribution and misdiagnosis false hit rates). These rates depend on the nature of the attacks and updates.

Let R be a fingerprinted relation with the embedded codeword $\Gamma = (f_0, \ldots, f_{L-1})$. We analyze the robustness of our fingerprint detection algorithm against representative attacks in the above attack classes. Unless otherwise stated, our analysis applies to all three fingerprinting schemes.

5.2 Bit-Flipping Attacks

In a bit-flipping attack, an attacker selects some bits and toggles their values. Now, the bit positions used by the fingerprint algorithms to embed and detect fingerprints are computed using a cryptographic hash function. We assume that an attacker does not know the secret key and so has no information about the values or positions of embedded bits. We also assume that he possesses a single fingerprinted copy of the data.

Suppose that the attacker toggles some randomly selected bits that are available for embedding fingerprint bits rather than for constructing virtual primary keys. This is a reasonable assumption since toggling some significant bits that are used in construction of virtual primary key would cause large data errors.

Let the attacker examine each fingerprintable bit independently and select it for toggling with probability p. Let $q = 1 - p$. We model bit flipping as Bernoulli trials (repeated independent trials) with probability p of success and q of failure (see Section 4.6).

Let the attacker apply the attack to an unmarked relation. Then, since the attacker toggles randomly selected bits, the resulting data will have the same characteristics as the original data for fingerprint detection. Thus, from Section 4.6, the misdiagnosis false hit rate is

$$N * \Pi_l B(\lfloor \tau w_l \rfloor; w_l, 0.5).$$

Now, let the attacker apply the attack to a fingerprinted relation. Consider the probability p_l that one particular fingerprint codeword bit f_l is destroyed. Now, each fingerprint bit f_l is actually embedded w_l times in R. For the detection algorithm to fail to recover the correct fingerprint bit, at least $(1 - \tau)w_l$ embedded bits that correspond to the fingerprint bit must be toggled (or, equivalently, more than $w_l - \lfloor \tau w_l \rfloor - 1$ bits must be toggled). Thus,

$$p_l = B(w_l - \lfloor \tau w_l \rfloor - 1; w_l, p)$$

The probability that the codeword bit is recovered correctly is $q_l = 1 - p_l$. Then, the probability that the entire codeword is recovered correctly is $\Pi_l q_l$. The probability that the codeword is not recovered correctly (i.e., the false miss rate) is $1 - \Pi_l q_l$. This false miss rate is bounded by $1 - (\min_l q_l)^L$, where $\min_l q_l = 1 - \max_l p_l \approx 1 - B(w_{min} - \lfloor \tau w_{min} \rfloor - 1; w_{min}, p)$.

Figures 1 and 2 plot the probability of a successful attack for different parameter values. The three parameter values which are varied are $w_{avg} = \sum_l w_l / L$ which is the average number of times

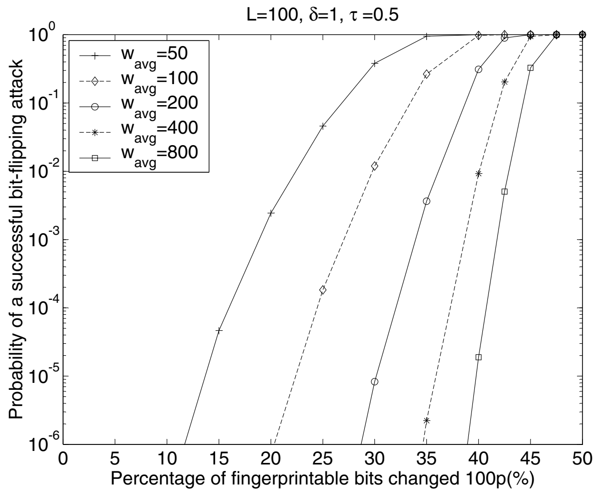

Figure 1: False miss rate under a bit-flipping attack

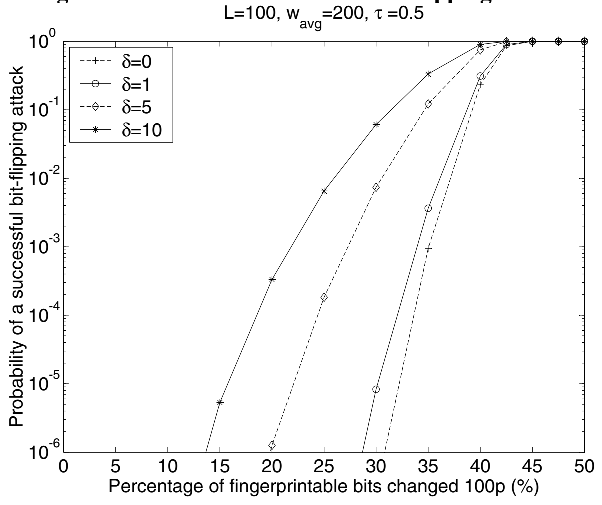

Figure 2: False miss rate under a bit-flipping attack

that a fingerprint bit is embedded into the relation, $\delta = (w_{max} - w_{min})/w_{min}$ which is the duplicate index, and p which is the flipping probability. The figure uses a value of 100 for L and 0.5 for τ. The probability of a successful attack (i.e., the false miss rate) depends on the distribution of w_l ($l = 0, \ldots, L-1$) which is assumed to be uniform in the interval[1] $[w_{min}, w_{max}]$ with mean w_{avg}.

Figure 1 shows that, for a fixed duplicate index δ, the larger the average number w_{avg} of fingerprint bit embeddings, the harder it is for bit-flipping attacks to succeed. When the average number of fingerprint bit embeddings is fixed, Figure 2 shows that a larger duplicate index renders the scheme more vulnerable to bit-flipping attacks. Both figures show that with a proper choice of parameters, a successful attack requires p to be large, causing a perceptible change to the relation. The figures also show that the average embedding rate w_{avg} does not need to be large to achieve this.

5.3 Subset Attacks

Consider a subset attack where the pirated data is a subset of tuples of a fingerprinted relation. Suppose that a relation has η tuples and that an attacker examines each tuple independently and selects it with probability $q' = \frac{\zeta}{\eta}$ for inclusion in the pirated relation. The pirated relation will thus have ζ tuples on average. The probability that a tuple is deleted is $p' = 1 - q'$.

Let the attack be applied to an unmarked relation. Fingerprint detection will deal with a relation with $\zeta = q' * \eta$ tuples on average. Then, from Section 4.6, the misdiagnosis false hit rate is

$$N * \Pi_l B(\lfloor \tau w'_l \rfloor; w'_l, 0.5)$$

where $w'_l = q' w_l$ is the number of times each fingerprint bit f_l would have been embedded in the relation had the relation been fingerprinted (assume w'_l is an integer). The misdiagnosis false hit rate still has an upper bound $\frac{N}{2L}$ if $w'_l > 0$ for all l.

Suppose that a subset attack is applied to a fingerprinted relation and that there is no other attack or benign update on the data. Then, for the attack to be successful, it must delete all embedded bits for at least one codeword bit. Now, each codeword bit f_l is embedded w_l times in the original relation, so the probability u_l that a codeword bit f_l is erased completely is:

$$u_l = B(w_l; w_l, p') = p'^{w_l}$$

Then, $v_l = 1 - u_l$ is the probability that a codeword bit f_l is detected, $\Pi_l v_l$ is the probability that the entire codeword is detected correctly, and $1 - \Pi_l v_l$ is the false miss rate.

Subset attacks have similar trends as bit-flipping attacks. Figure 3 shows that, for fixed duplicate index, the larger the average number w_{avg} of fingerprint bit embeddings, the harder it is for subset attacks to succeed. When the average number of fingerprint bit embeddings is fixed, Figure 4 shows that a larger duplicate index renders the scheme more vulnerable to subset attacks.

5.4 Superset Attacks

Consider a superset attack where the pirated data is a superset of tuples of a fingerprinted relation. Statistically, for those newly added tuples, we have 50% probability to detect correct bits (purely by chance). If we set $\tau = 0.5$ in fingerprinting, the superset attack will never succeed.

5.5 Attribute Attacks

Consider an attack that *adds* some new attributes into a fingerprinted relation. Since T-scheme depends on an ordering over the attributes, it must relate the new relation's attributes to the original relation's attributes and recover the original ordering. E-scheme is unaffected since fingerprint detection is element-based. As with superset attacks, adding attributes will never succeed if $\tau = 0.5$.

M-scheme is affected little by the addition of attributes. Recall that M-scheme selects two hash values, say h_1 and h_2, in $\{|\mathcal{H}_1(\mathcal{K}, r.A_i^{sig})| : i = 0, \ldots, \nu - 1\}$ that are closest to zero in construction of the virtual primary key, and marks attribute(s) A_i if its(their) hash value $|\mathcal{H}_1(\mathcal{K}, r.A_i^{sig})|$ is closest to zero. Consider that one attribute A_x is added. The embedded fingerprint can be extracted correctly if the hash value $|\mathcal{H}_1(\mathcal{K}, r.A_x^{sig})|$ does not become one of the two hash values h_1 and h_2. Theoretically, if the distribution of hash values $|\mathcal{H}_1(\mathcal{K}, r.A_x^{sig})|$ is uniform from zero to U (here U could be the largest integer in a computer system), the probability that the hash value $|\mathcal{H}_1(\mathcal{K}, r.A_x^{sig})|$ becomes one of the two hash values is $\max(h_1, h_2)/U$.

If an attacker *deletes* some attributes from a fingerprinted relation, a similar situation holds. T-scheme must recover the original ordering over the attributes; tuples in which the deleted attributes were marked must be considered deleted. For E-scheme, attribute deletion can be analyzed the same way as tuple deletion (see subsection 5.3). For M-scheme, fingerprint detection for some tuples

[1] w_{min} and w_{max} can be computed from the two parameters w_{avg} and δ: $w_{min} = \frac{2w_{avg}}{2+\delta}$ and $w_{max} = \frac{2(1+\delta)w_{avg}}{2+\delta}$.

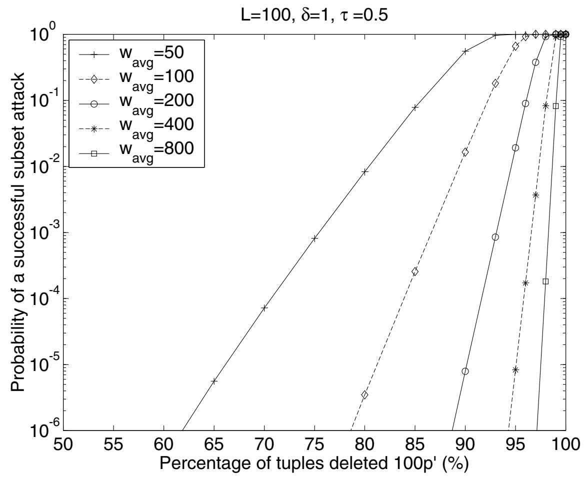

Figure 3: False miss rate under a subset attack

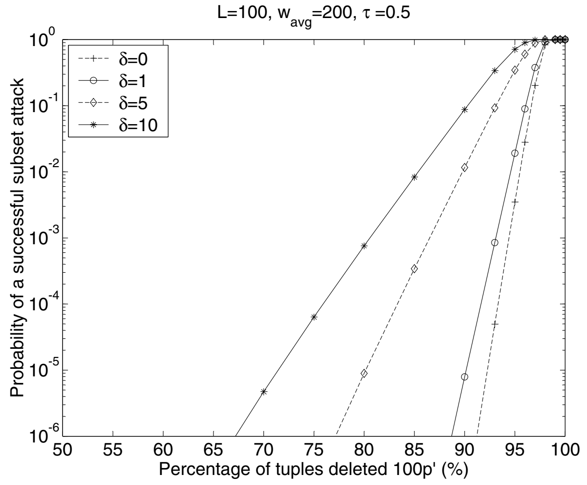

Figure 4: False miss rate under a subset attack

r is affected only if the deleted attributes in r have hash values that are used in the construction of the virtual primary key.

If an attacker *modifies* some attributes from a fingerprinted relation, the order of attributes does not change. T-scheme and E-scheme can be analyzed the same way as in bit-flipping attacks. M-scheme can be analyzed the same way as in attribute deletion.

Section 7 presents experimental results on attribute attacks. A detailed statistical analysis will be provided in an extended version of this paper.

5.6 Collusion Attacks

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 at all.

A well-known solution to the collusion problem was proposed by Boneh and Shaw [2], and many other solutions have been proposed subsequently (e.g., [5]). These schemes focus on the choice of codewords used by a fingerprinting scheme. They show that by proper choice of codewords (usually very long), fingerprinting schemes can be made collusion secure. Most of these schemes do not address the problem of inserting or detecting fingerprints in data, but rather apply to any embedding schemes that satisfy two properties. First, an attacker can only detect that a bit position was used during fingerprint insertion if the attacker has data copies that differ in value at that position. Second, although an attacker may determine that a particular data bit was used to embed some codeword bit, the attacker cannot determine which codeword bit it represents.

The fingerprinting schemes discussed in this paper satisfy these properties since both the locations of embedded bits and the relationship between codeword bits and embedded bits are hidden using a keyed pseudorandom function. Thus, we can use any of those collusion-secure codeword schemes by replacing the hash function $\mathcal{H}_0$ and the subroutine *detect* in the fingerprinting algorithms. The resulting fingerprinting schemes will have the collusion-secure properties of the codeword scheme. This has been demonstrated for T-scheme [6] using an adapted version of Boneh and Shaw's algorithm. The same arguments also apply to E-scheme and M-scheme.

5.7 Invertibility Attacks

An invertibility attack [3] discovers a fictitious secret key that extracts a valid mark from pirated data. A pirate can use the discovered key to claim legitimate ownership of the data. Alternately, a pirate can claim innocence by claiming that a merchant used this attack to obtain evidence of piracy. Since our marking schemes are all symmetric [7], invertibility attacks are only relevant if the schemes are used to embed watermarks.

Consider an invertibility attack where an attacker randomly selects a secret key. The key may cause the fingerprint detection algorithm to extract a random codeword from the pirated relation. The probability of this is the same as the misdiagnosis false hit rate:

$$N * \Pi_l B(\lfloor \tau w_l \rfloor; w_l, 0.5)$$

An attacker may choose parameters γ, L, τ, and N to increase his probability of success. In particular, if he selects $\tau = 0.5$, this probability reduces to $\frac{N}{2^L}$. This attack can be thwarted by requiring (in convention or standard) long fingerprints.

5.8 Additive Attacks

In an additive attack, a traitor inserts another mark before distributing a pirated database. A traitor may insert a watermark to claim ownership of the database and he may insert a fingerprint to claim that the database was provided to a user legitimately. This type of attack is discussed in [1] in the context of watermarking; the solution they propose is applicable to our fingerprinting schemes as well.

6. DISCUSSION

Note that a database relation is not always fingerprintable due to the following reasons. First, if $\eta < \gamma' L$ or $\eta < L$, then the relation does not have enough tuples to embed a fingerprint. Collusion-secure codes are usually very long and hence are only suitable for large databases. Other codes are not problematic since their lengths are usually several orders of magnitude smaller than the typical sizes of databases. For example, a fingerprint length of hundred bits is good enough for practical use provided that collusion attacks are not a threat. Second, if the duplicate index is infinity, then at least one fingerprint bit cannot be embedded into data and the embedded fingerprint is not detectable. The duplicate index can be lowered either by decreasing ξ or γ' or by using more attributes in construction of the virtual primary key. Third, if varying L, τ and other fingerprinting parameters is insufficient to yield low enough false hit rate and false miss rate, then fingerprint detection is not trustworthy.

For fingerprintable database relations, a selection of fingerprinting schemes depends on the primary key and the duplicate index. In the case that the relation has a primary key attribute and the primary key is not expected to be changed in an attack, T-scheme is the best choice because it tends to yield the lowest duplicate index. Roughly speaking, the lower the duplicate index, the more robust a fingerprinting scheme.

If a database relation has no primary key attribute or the primary key might be changed in an attack, we need to choose between E-scheme and M-scheme. Because E-scheme tends to yield the largest duplicate index, we often select M-scheme over E-scheme. However, in the case that the duplicate indices of these two schemes are the same, E-scheme could be a better choice because it supports element-level updatability.

7. EXPERIMENTS

The experiments were performed on a Dell OPTIPLEX GX260 computer with the Windows 2000 operating system, 2 GHz Intel Pentium 4 processor, 512 MB of memory, and a 40 GB hard disk drive. We used the real-life Forest Cover Type dataset, available from the University of California-Irvine KDD Archive (`http://kdd.ics.uci.edu/databases/covertype/covertype.html`). The dataset has 581,012 tuples, each with 61 attributes and no primary key. We chose the first ten integer-valued attributes as candidates for fingerprinting. We focused on M-scheme unless otherwise stated.

Figure 5 illustrates the impact that fingerprinting has on the mean and variance of the values of marked attributes. Blank entries in the table indicate that the variance of the attribute was unchanged. The mean of each attribute's values was unchanged when rounded to the nearest integer. The minuscule change in these statistics validates our assertion that fingerprint insertion is imperceptible. In our experiments, $1/\xi'$ is the fraction of the least significant bits in $r.A_i$ that are used for embedding a fingerprint. Note that ξ' and ξ are negatively correlated.

Figure 6 shows the change of duplicate index as a function of ξ' and γ (fraction of tuples used for fingerprinting). The larger the parameter ξ', the less the duplicate index δ. As ξ' increases, more of the most significant bits are used in the construction of virtual primary keys, and the likelihood of generating duplicate virtual primary keys decreases.

Figure 7 compares M-scheme with T-scheme and E-scheme in terms of the duplicate index. For T-scheme, we added an extra attribute called *id* to serve as the primary key. Due to the uniqueness of such a primary key, the duplicate index of T-scheme is closest to zero compared with that of M-scheme and E-scheme. On the other hand, the duplicate index of E-scheme is always infinity, indicating that E-scheme cannot be used to fingerprint this relation.

Figures 8 and 9 report the performance of M-scheme under subset attacks and attribute deletion attacks. We used the following parameters $\nu = 10, \xi' = 4, L = 58, \tau = 0.5$ and $N = 1000$ (N has no effect on these results) and tested different γ values. For each value of gamma ($\gamma = 12, 25, 50, 100$) and tuple percentage (1-10%), we randomly selected the percentage of tuples from the fingerprinted dataset and tested whether the detection algorithm returned the correct fingerprint. We repeated this test 100 times and calculated the percentage of fingerprints that were detected correctly. The attribute deletion test was performed analogously.

In our experiments, fingerprint insertion took about 36 seconds and fingerprint detection took about 13 seconds on average. In the worst case where $\gamma = 1$ (i.e., each tuple was designed to be modified), fingerprint insertion took about 42 seconds and detection took about 14 seconds.

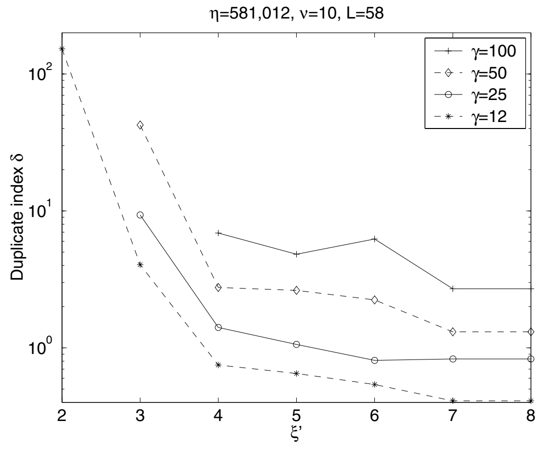

Figure 6: Change of duplicate index

γ	δ		
	T-scheme	**M-scheme**	**E-scheme**
100	0.58	6.89	infinity
50	0.61	2.70	infinity
25	0.14	1.41	infinity
12	0.07	0.75	infinity

Figure 7: Duplicate index for different fingerprinting schemes ($\xi' = 4$)

8. CONCLUSION

We have presented schemes for embedding and detecting fingerprints in database relations without relying on primary keys. These schemes can mark relations that have no primary keys and relations which retain value even if their primary keys are altered in attacks. We identified the duplicate and deletion problems in fingerprinting such database relations and proposed techniques for constructing virtual primary keys to mitigate these problems.

An interesting extension to this work is to optimize the construction of virtual primary keys. The goal is to maximize the robustness of the fingerprinting scheme for a fixed set of fingerprinting parameters. Future work also includes extending our embedding scheme so that it marks both non-numeric and numeric attributes, and developing an asymmetric version of our scheme.

9. REFERENCES

[1] R. Agrawal and J. Kiernan. Watermarking relational databases. In *Proceedings of VLDB*, 2002.

[2] D. Boneh and J. Shaw. Collusion secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

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

[4] D. Gross-Amblard. Query-preserving watermarking of relational databases and XML documents. In *Proceedings of*

			$\gamma = 100$			50			25			12		
Attribute	**Mean**	**Variance**	$\xi' = 2$	4	8	2	4	8	2	4	8	2	4	8
Elevation	2959	78391	+2			+2			-17					
Aspect	156	12525												
Slope	14	56												
Horz-Dist-To-Hydrology	269	45177												
Vert-Dist-To-Hydrology	46	3398												
Horz-Dist-To-Roadways	2350	2431272	-34			-25			-24			+4	-1	
Hillshade-9am	212	717												
Hillshade-noon	223	391												
Hillshade-3pm	143	1465												
Horz-Dist-To-Fire-Points	1980	1753490	+5			+11	-2		+28	-2		+3		

Figure 5: Change in variance introduced by fingerprinting (After rounding to the nearest integer, the mean does not change)

the twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pages 191–201. ACM Press, 2003.

[5] H. Guth and B. Pfitzmann. Error and collusion secure fingerprinting for digital data. In *Information Hiding '99, LNCS 1768, Springer-Verlag*, pages 134–145, 2000.

[6] Y. Li, V. Swarup, and S. Jajodia. Fingerprinting relational databases. Technical report, Center for Secure Information Systems, George Mason University, Fairfax, VA, May 2003.

[7] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 151–160, 1997.

[8] R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 98–109, 2003.

[9] N. Tran. Hiding functions and computational security of image watermarking systems. In *15th IEEE Computer Security Foundations Workshop*, pages 295–306, 2002.

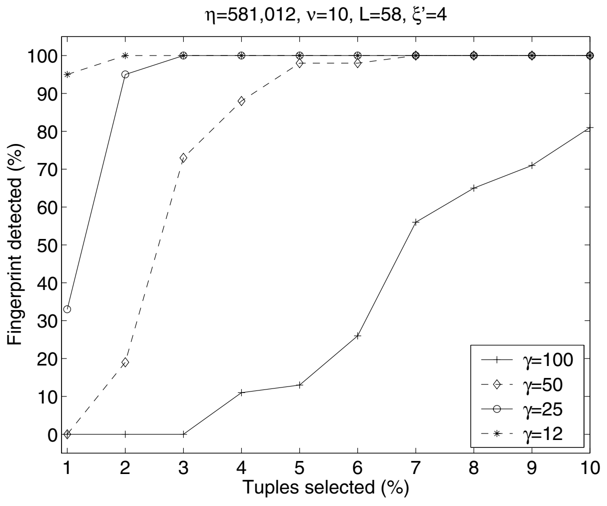

Figure 8: Fingerprint detection under subset attacks

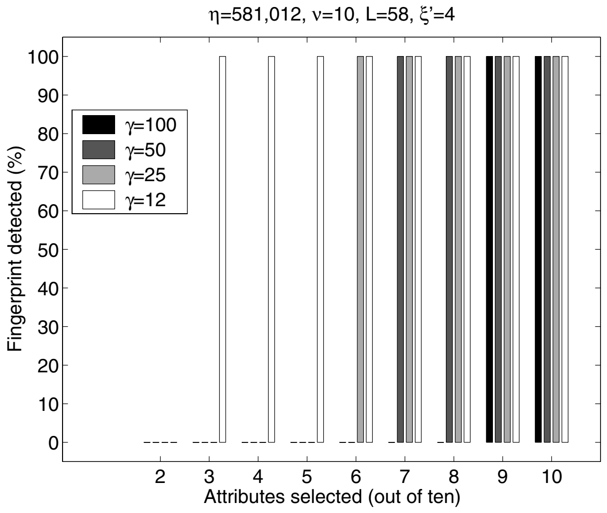

Figure 9: Fingerprint detection under attribute deletion attacks