12-2005

A Block Oriented Fingerprinting Scheme in Relational Database

Siyuan LIU
Peking University

Shuhong WANG
Peking University

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Weizhong SHAO
Peking University

DOI: https://doi.org/10.1007/11496618_33

A Block Oriented Fingerprinting Scheme in Relational Database

Siyuan Liu[1,3], Shuhong Wang[2,4], Robert H. Deng[4], and Weizhong Shao[1]

[1] Institute of Electronics Engineering & Computer Science,
Peking University (PKU), China 100871
iceice@cs.pku.edu.cn, wzshao@pku.edu.cn
[2] School of Mathematical Sciences, PKU, China 100871
wshong@math.pku.edu.cn
[3] Institute for Infocomm Research, Singapore 119613
[4] School of Information Systems,
Singapore Management University, Singapore 259756
robertdeng@smu.edu.sg

Abstract. The need for protecting rights over relational data is of ever increasing concern. There have recently been some pioneering works in this area. In this paper, we propose an effective fingerprinting scheme based on the idea of block method in the area of multimedia fingerprinting. The scheme ensures that certain bit positions of the data contain specific values. The bit positions are determined by the keys known only to the owner of the data and different buyers of the database have different bit positions and different specific values for those bit positions. The detection of the fingerprint can be completed even with a small subset of a marked relation in case that the sample contains the fingerprint. Our extensive analysis shows that the proposed scheme is robust against various forms of attacks, including adding, deleting, shuffling or modifying tuples or attributes and colluding with other recipients of a relation, and ensures the integrity of relation at the same time.

Keywords: Fingerprinting, Scheme, Block, Security.

1 Introduction

1.1 Background

Due to the rapid development and widespread use of digital assets, such as software, images, video, audio and text, protection of ownership of digital content is increasingly being a matter of great concern. There are many methods to prevent piracy of digital content and fingerprinting, a special type of information hiding technique, is a very promising one. Consider a scenario where merchants sell digital data to buyers. Some dishonest buyers may redistribute the data to others without permission from the merchants. A merchant may use a fingerprint scheme to embed a buyer-specific mark into a data copy and subsequently detect the mark in pirated data and use the mark to identify the traitor who distributed

C. Park and S. Chee (Eds.): ICISC 2004, LNCS 3506, pp. 455–466, 2005.

the data. Fingerprinting is often discussed in comparison or extension to watermarking. Watermarking is another type of information hiding technique 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.

Till now, the study of fingerprinting and watermarking has focused mainly on multimedia content which includes digital images, audio and video. There has recently been some pioneering research presented in [1,2,3] and more recent work of [7,8] in the area of protecting relational database. In [1], the authors present the first known database watermarking technique that marks the numeric attributes of relational data. The algorithm uses a hash function depending on a private key known only to the owner. The hash function decides the tuples, attributes within a tuple, and bit positions within an attribute to be marked. Only when the attackers have access to the private key, can they detect the watermark with a high probability. The technique survives several attacks and preserves mean and variance of all numerical attributes. In [2], the authors generalized the watermarking technique in [1] to enable the fingerprinting of relational data. The fingerprinting technique enables a buyer-specific bit string to be embedded and extracted from a relational database, as compared to the watermarking technique which enables a single watermark bit to be embedded and extracted from a relational database. In [3], the authors extend the technique in [1,2] which is dependent on primary keys and construct a virtual primary key scheme for relational databases which do not have primary keys. The schemes in [1,2,3] are robust against various attacks including flipping bits, adding or deleting tuples and guessing secret keys.

1.2 Related Work

In [4], the authors propose a block oriented fingerprinting scheme in spatial domain, which inspires us very much. The scheme first produces one fingerprint for every buyer and then divides the image to be fingerprinted into a number of blocks of size $\beta \times \beta$ and the number of blocks m is equal to $\frac{ht(I) \times wd(I)}{\beta \times \beta}$ ($ht(I)$ and $wd(I)$ are the height and width of the image in pixels, respectively). Then the scheme permutes the blocks in an order which is specific for every buyer. The permutation and the information of the buyer are both stored in a database known to the merchant only. Then for every block the scheme calculates the minimum and maximum intensities of the pixels in the block, and according to the corresponding bit of the fingerprint, increases intensities of the pixels in the block if the bit is 1 or decreases the intensities if the bit is 0. So every buyer will get one marked image which is different for everyone.

1.3 Our Contribution

In this paper, we propose a novel and flexible scheme to fingerprint a relational database based on the block method for fingerprinting an image as described

above. Compared to the previous works, our scheme has three novel features. First, it is based on the block method so that the owner can change the size of the blocks and change the degree of distortion to the database. Secondly, our scheme has low distortion introduced to the data values without compromising the integrity (mean and variance) of the data. As will be seen later, our analysis show that our scheme is resistant to many attacks including collusion attacks. Third, our scheme can be applied to databases with primary keys and with a little extension to databases without primary keys.

The scheme we propose is inspired by the method described above, but is very different from it. Because there exist some fundamental differences between the characteristics of multimedia data and relational data so that we can not carry the multimedia techniques over to the realm of relational database. The differences include:

- It generally does not cause perceptual changes in the object to drop or replace portions of a multimedia object. However, the pirate of a relation can frequently delete, insert or replace the database tuples.
- Multimedia objects consist of a large number of bits, with considerable redundancy, thus providing a larger cover to hide information. A database relation consists of tuples, each of which represents a separate object. The fingerprint needs to be spread over these separate objects.
- The relative spatial/temporal positions of various pieces of a multimedia object is often fixed, but it is not the case for the tuples of a relation database.
- There are many psycho-physical phenomena based on the human visual system and human auditory system which can be exploited for mark embedding. However, one can not exploit such phenomena in relational databases.

Due to the differences, our fingerprinting scheme for relational database is only inspired by the block fingerprinting method for images, and is very different from it.

The paper is structured as follows. Section 2 describes an effective fingerprinting scheme based on the block method. Section 3 analyzes the security of the proposed scheme. The conclusion is given in Section 4.

2 An Effective Fingerprinting Scheme

Let's consider the following scenario. Alice is the owner of a relational database which is sold to many buyers. Later Alice found that someone owned the database but she never sold it to him. She needs certain methods to detect who distributed the database illegally. In this section, we will describe an effective fingerprinting scheme which can be used to embed a fingerprint into the database and detect it when necessary. We assume that the relational database has a primary key. The scheme can be extended for relational databases without primary keys based on the technique in [3].

2.1 Requirements

A fingerprinting scheme should satisfy the following properties.

- Detectability: Alice should be able to detect the fingerprint by examining limited tuples from a suspicious database. The suspicious database may be only a small part of the fingerprinted database or a modified version of the fingerprinted database.
- Imperceptibility: Modifications caused by fingerprinting should not reduce the usefulness of the database. In addition, commonly used statistical measures such as mean and variance of the numerical attributes should not be significantly affected.
- Robustness: A fingerprint scheme should be robust against benign database operations and malicious attacks that may destroy or modify embedded fingerprints. Benign operations include adding tuples, deleting tuples, and updating tuples in database relations. Malicious attacks include selective modifications of fingerprinted relations, taking subsets of relations, and modifying or erasing the embedded fingerprint. These common attacks have been identified in [1,2] and described in the following.

  1. Randomization attacks: Certain bits of a fingerprinted database are assigned random values so that some fingerprint bits may not be detected.
  2. Zero out attacks: Values of some bits of fingerprinted database are changed to zero which results in that the fingerprint can not be detected correctly.
  3. Bit flipping attacks: Values of some bits of fingerprinted database are inverted thus the fingerprint can not be detected correctly.
  4. Rounding attacks: Some bits of fingerprinted database are deleted due to the rounding of numerical values so that the fingerprint may not be detected correctly.
  5. Subset attacks: A subset of tuples or attributes of a fingerprinted relation appear in a pirated database so that the fingerprint can not be detected correctly.
  6. Superset attacks: Some new tuples or attributes are added to a fingerprinted database, which can affect the correct detection of the fingerprint.
  7. Additive attacks: Adding an additional fingerprint to a pirated copy thus to confuse a third party.
  8. Invertibility attacks: Discovering a fictitious fingerprint in a relation thus confusing the owner.
  9. Majority attacks: Creating 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 so that the owner can not detect the fingerprint.
  10. Mix and match attacks: Creating a pirated copy by combining subsets of tuples and attributes from each fingerprinted copy so that the owner can not detect the fingerprint.

The first four types of attacks reduce the accuracy of data. The following two classes of attacks modify relations but didn't reduce accuracy. The seventh and eighth types of attacks seek to provide a traitor or pirate with evidence that raises doubts about a merchant's claims. The last two types of attacks are collusion attacks which require attackers to have access to multiple fingerprinted copies of the same relation but with different embedded fingerprints.

2.2 Notation and Parameters

Consider a database relation R that has a single primary key attribute P and v numerical attributes $A_0, \ldots, A_{v-1}$. Without loss of generality, let the schema of R be $R(P, A_0, \ldots, A_{v-1})$ and let the database has 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 mark 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 on 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 $\Gamma = (f_0, \ldots, f_{L-1})$ is a binary string with length $L \geqslant logn$. Each user is assigned a unique fingerprint of the same length L. A fingerprint is embedded into each copy of R and the fingerprinted data is then distributed to the corresponding user.

User i's fingerprint is computed by a cryptographic hash function H_0 whose input is the concatenation of a secret key K(known by the merchant only) and user identifier ID_i. The output of H_0 is a binary string of length L. We shall assume that this results in a unique fingerprint for each user $i = 0, \ldots, n-1$. This is usually the case when $L > logn$ 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. We use one pseudo-random producer, which can be the BBS producer, to produce a series of random numbers, and every user have one different threshold for the pseudo-random producer. We also use one cryptographic hash function H_1:

$$H_1(K, ID_i) = H(K \| H(K \| ID_i)) \tag{1}$$

where H is a standard hash function(e.g., MD5 or SHA), and $\|$ denotes concatenation. Table 1 gives the notations we use.

2.3 Insertion Stage

At the stage of fingerprint insertion, we first regard the bits of the attributes that can be used to embed the fingerprint bits as a two-dimension image. For example, Table 2 gives a small part of a relational database. P is the primary key, the last three bits of A_1 and A_2 can be used to embed fingerprint bits. We first extract the three least significant bits of A_1 and A_2 and combine them together as shown in Table 3. Then we divide Table 3 into 6 parts each of size $\beta \times \beta$ (here $\beta = 2$), as given in Table 4.

Table 1. The notions

Notations	Meaning
ν	The number of attributes in the relational database that can be marked
η	The number of tuples in the relational database
ξ	The number of the least significant bits that can be used for marking in an attribute
β	The size of every block
n	The number of users

Table 2. Part of a relational database

P	A_1	A_2
1	01100011	00001001
2	10000010	00100111
3	01001111	10010001
4	00000000	00000101

Table 3. The bits available for fingerprinting

011001
010111
111001
000101

Table 4. The 6 2×2 blocks

01	10	01
01	01	11
11	10	01
00	01	01

Table 5. The insertion algorithm

```
1. for one buyer
2.    produce the fingerprint Γ for the buyer
3.    choose one threshold r_0 for the pseudo random number generator
4.    divide the database attributes bits into blocks of size β × β
5.    i=0,j=0
6.    for each block B_i
7.       r_1 =random(r_0)
8.       x = H_1(r_1, ID) mod β
9.       r_2 =random(r_1)
10.        y = H_1(r_2, ID) mod β
11.       B_i(x, y) = B_i(x, y) ⊕ f_j
12.       r_0 = r_2
13.       i++,j++ if j==L then j=0
14.    end for
15. end for
```

Now we use the pseudo random generator to produce a random number r, and according to the result of $H_1(r, ID)$ mod β to decide where the fingerprint bit should be embedded. Table 5 gives the insertion algorithm.

Table 6. The detection algorithm

1. sort S to S' according to the primary key
2. divide bits in S' into blocks of size $\beta \times \beta$
3. for each buyer, retrieve the corresponding r_0:
4. for each block B_i
5. $r_1 =$random(r_0)
6. $x = H_1(r_1, ID) \bmod \beta$
7. $r_2 =$random(r_1)
8. $y = H(r_2, ID) \bmod \beta$
9. $F_i = S'(B_i(x,y)) \oplus R(B_i(x,y))$ if $S'(B_i(x,y))$ is in S
10. $r_0 =$random(r_2)
11. i++
12. end for
13. end for
14. for each buyer, retrieve his fingerprint $\Gamma = (f_0, \ldots, f_{L-1})$
15. define $f'_i = 1, 0 \le i \le L-1$, if $\frac{\#\{k \mid F_{i+kL}=1, 0 \le i+kL \le m-1\}}{\omega} \geqslant \tau$, otherwise define $f'_i = 0$.
16. if $\Gamma = \Gamma' = (f'_0, \ldots, f'_{L-1})$, the data is said to had distributed by this buyer.

Line 7 and line 9 use a pseudo random number generator to produce a random number, respectively. Line 8 and line 10 determine the position in a block the fingerprint bit should be embedded. Line 13 means that the next fingerprint bit is ready to be embedded and the next block is ready to be marked. When all the fingerprint bits have been embedded and the blocks are not over the fingerprint bits will be embedded again until all the blocks have been used.

2.4 Detection Stage

In the fingerprint detection stage, the merchant first sorts the suspicious database S according to the primary key. If there are some tuples deleted, they are added as in the unfingerprinted original data according to the primary key. Then divide the bits that can be used to embed fingerprint bits into blocks of size $\beta \times \beta$ and mark the blocks which are included in R but not in S. Comparison to the original blocks which don't contain fingerprint, the merchant decides whether the suspected relational database is pirated or not. Table 6 gives the fingerprint detection algorithm.

Line 9 means that for this buyer, the i-th block is detected being marked with F_i. $S(B_i)$ is the i-th block in the suspicious database and $R(B_i)$ is the i-th block in the original database. In line 15, m is the number of blocks, ω times is the number of times a fingerprint has been embedded in the database. If $\lceil \tau\omega(\tau \in [0.5, 1]) \rceil$ detected bits are 1, the detected fingerprint bit is said to be 1, otherwise is said to be 0.

2.5 Fingerprinting Relational Databases Without Primary Keys

The fingerprinting scheme described above is predicated on the assumption that the relational database have a primary key. And our scheme can be easily ex-

tended to be used for relations that are without primary keys based on the techniques proposed in [3].

3 Robustness

In this section, we analyze the robustness of our fingerprint algorithm against representative attacks under the assumption that the attackers don't change the values of primary keys. In addition, we also investigate *false hit rate*, the probability of failing to detect an embedded codeword correctly. Let R be a fingerprinted relation with the embedded fingerprint $\Gamma = (f_0, \ldots, f_{L-1})$.

3.1 Some Discussion

An important parameter in our scheme is β, which determines the size of the block. If β is large, the number of blocks reduces and the number of times a fingerprint being embedded also reduces. But if β is too small, and the number of buyers is more than the size of the block, the situation will happen that different buyers' fingerprint is embedded into the same bit position in the same block. To avoid the situation, the size of the block should be more than the number of buyers, meaning that β should be at least more than $\sqrt{n}$. On the other hand, if β is too big, the fingerprint can be embedded for only a small number of times, the correctness of the detection may be impaired. The times that the fingerprint can be embedded can be computed as $\omega = \lfloor \frac{\xi\eta\nu}{\beta\times\beta\times L} \rfloor$. It is to say that every fingerprint bit f_l is embedded ω times

3.2 Cumulative Binomial Probability

We use Bernoulli trials in our robustness analyze. Repeated independent trails are called Bernoulli trials if there are only two possible outcomes for each trial and their probabilities remain the same throughout the trials. Let $b(k; n, p)$ be the probability and n Bernoulli trials with probabilities p for success and $1 = 1-p$ for failure result in k successes and $n - k$ failures. Then

$$b(k; n, p) = \binom{n}{k} p^k q^{n-k} \tag{2}$$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3}$$

Denote the number of successes in n trials as S_n. The probability of having at least k successes in n trials, the cumulative binomial probability, can be written as

$$P\{S_n \geqslant k\} = \sum_{i=k}^{n} b(i; n, p) \tag{4}$$

For brevity, define

$$B(k; n, p) = \sum_{i=k}^{n} b(i; n, p) \tag{5}$$

3.3 Bit-Flipping Attacks

In a bit-flipping attack, an attacker selects some bits and toggles their values. In our scheme, the bit positions used for fingerprinting are computed by a pseudo-random generator which has a threshold known only to the merchant and a cryptographic hash function. We assume that the attacker does not know the threshold so that he has no information about the values or positions of embedded bits. We also assume that the attacker possesses a single fingerprinted copy of the data. Now, let the attacker examine each bit available for fingerprinting independently and select it for flipping with probability p. Let $q = 1-p$. We model bit flipping as Bernoulli trials with probability p of success and q of failure. 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. Each fingerprint bit f_l is actually embedded ω times as discussed above. For the detection algorithm to fail to recover the correct fingerprint bit, at least $(1-\tau)\omega$ embedded bits that correspond to the fingerprint bit must be changed. It also can be said that more than $\omega - \lceil \tau\omega \rceil + 1$ bits must be changed. So $p_l = B(\omega - \lceil \tau\omega \rceil + 1; \omega, p)$. The probability that the codeword bit is recovered is $q_l = 1 - p_l$. Then the probability that the entire codeword is recovered correctly is $\prod_l q_l = (1-p_l)^L$. And *the false hit rate* is $1-(1-p_l)^L$. Table 7 describes the probability of a successful attack for different parameter values. Here we set $\eta = 100000, \xi = 3$, $\nu = 3$, $\tau = 0.5$ and $L = 100$. We can see that when β is increasing and p is less than 40% the probability for a successful bit-flipping attack is also increasing for the same p. And when p is more than 40%, the probability is decreasing while β is increasing. So we can choose the appropriate β. For example, $\beta = 30$ is adapt to the situation when the length of fingerprint is 100 bits. Comparison to the scheme in [2], there is some development in our scheme on robustness against bit-flipping attacks.

Table 7. The probability for a successful bit-flipping attack for different block sizes

	$p = 10\%$	$p = 20\%$	$p = 30\%$	$p = 40\%$	$p = 50\%$
$\beta = 5$	0	0	0	0	0
$\omega = 360$	0	0	0	0	0
$\beta = 10$	0	5.0610×10^{-9}	2.2564×10^{-3}	0.8837	1.0000
$\omega = 90$	0	8.3235×10^{-8}	1.2954×10^{-2}	0.9962	1.0000
$\beta = 15$	1.9597×10^{-9}	5.0260×10^{-4}	0.2151	0.9996	1.0000
$\omega = 40$	1.6237×10^{-7}	8.4818×10^{-3}	0.7743	0.9999	1.0000
$\beta = 20$	2.4620×10^{-5}	3.4324×10^{-2}	0.7567	0.9999	1.0000
$\omega = 22$	2.0701×10^{-3}	0.4599	0.9999	1.0000	1.0000

Table 7 also gives the comparison result on the probability for a successful bit-flipping attack for different ω based on the scheme in [2]. For every cell, the first line is the probability for our scheme and the second line is the probability for the scheme in [2].

3.4 Subset Attacks

Consider a subset attack where the pirated data is a subset of tuples of a fingerprinted relation. Note that a relation has η tuples and that an attacker examines each tuple independently and selects it with probability q' for inclusion in the pirated relation. The pirated relation will thus have $\zeta = q'\eta$ tuples on average. The probability that a tuple is deleted is $p' = 1 - q'$. 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 at least $\lfloor \omega - \tau\omega \rfloor$ embedded bits for some codeword bit. Now, each codeword bit f_l is embedded ω times in the original relation, so the probability μ_l that a codeword bit f_l is erased completely is $\mu_l = B(\lfloor \omega - \tau\omega \rfloor; \omega, p')$. Then, $v_l = 1 - \mu_l$ is the probability that a codeword bit f_l is detected, $\prod_l v_l$ is the probability that the entire codeword is detected correctly, and $1 - \prod_l v_l$ is the false miss rate.

Table 8 shows the probability of a successful attack for different parameter values. Here we set $\eta = 100000, \xi = 3$, $\nu = 3$, $\tau = 0.5$ and $L = 100$. We can see that when β is increasing the probability for a successful subset attack is also increasing for the same p'. So we can change β to adapt to different needs.

Table 8. The probability for a successful subset attack for different block sizes

	$p' = 10\%$	$p' = 20\%$	$p' = 30\%$	$p' = 40\%$	$p' = 50\%$
$\beta = 5$	0	0	0	0	0
$\beta = 10$	0	2.09722×10^{-8}	5.5264×10^{-3}	0.9706	1.0000
$\beta = 15$	1.8719×10^{-8}	2.1669×10^{-3}	0.4660	0.9999	1.0000
$\beta = 20$	2.4596×10^{-4}	0.1471	0.9808	0.9999	1.0000

3.5 Attribute Attacks

If an attacker adds one new attribute into a fingerprinted relation. Because our detection algorithm first reassorts the suspected relation, the new attribute can be omitted.

If an attacker delete some attributes from a fingerprinted relation, tuples in which the deleted attributes were marked can be regarded as deleted. The situation can be analyzed as tuple deletion described in section 3.4.

If an attacker modifies some attributes from a fingerprinted relation, the situation can be analyzed as the bit-flipping attacks.

3.6 Collusion Attacks

Fingerprinting schemes are susceptible to collusion attacks by coalitions with access to multiple fingerprinted copies of the relation but with different embedded fingerprints. The attackers can create a useful data copy that does not implicate anyone of the attackers. During fingerprint detection, the copy may yield the fingerprint of an innocent buyer, or it may not yield a valid fingerprint at all.

There are many solutions to the collusion problem, a well-known of which was proposed by Boneh and Shaw [5], and many others have been proposed such as [6]. These solutions focus on the choice of codewords used by a fingerprinting scheme. They show that by a proper choice of codewords, fingerprinting schemes can be made collusion secure. Most of the solutions need the fingerprinting satisfy two properties, one is that 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 and the other is that 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 described in the paper satisfy these properties because the positions the codeword is embedded are hidden using a pseudo random function. And because β is known to the owner, the attacker do not know the relationship between the embedded bits and the codeword bits. On the other hand, different buyers' codewords are embedded in different locations, which also increase the difficulties an attacker can destroy the fingerprint. Thus, we can use any of those collusion-secure codeword schemes by replacing the hash function H_0 in the fingerprinting algorithms. This has been demonstrated in [2] using an adapted version of Boneh and Shaw's algorithm. It's the same for our schemes.

3.7 Additive Attacks

In an additive attack, an attacker may insert 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.

3.8 Distortion to Integrity and Consistency

In our scheme, the bits ready for embedding fingerprint bits are the least significant bits of the candidate attributes. The bits are randomly chosen for embedding fingerprint bits, which means that it is impossible for the bits are mostly embedded in only one or few attributes. So the changes of the mean and variance of the candidate attributes are almost imperceptive, which ensure the integrity and consistency of the relation.

4 Conclusion

In this paper, we have presented the scheme for embedding and detecting fingerprints in relational databases based on a block method. In addition, we have presented security analysis to show the robustness of our technique against various attacks.

For future work, we would like to optimize the detection process and investigate the possibility of extending our embedding scheme for both non-numeric and numeric attributes.

References

1. R.Agrawal and J.Kiernan. Watermarking relational databases. Proc. 28th International Conference on Very Large Data Bases (VLDB 2002), 2002.
2. Y.Li, V.Swarup, S.Jajodia. Fingerprint relational databases. Technical report, Center for secure Information Systems, George Mason University, Fairfax, VA, May 2003.
3. Yingjiu Li, Vipin Swarup, Sushil Jajodia. Construting a virtual primary key for fingerprinting relational data. DRM 03.
4. Tanmoy Kanti Das, Subhamoy Maitra. A robust block oriented watermarking scheme in spatial domain. 4the International Conference(ICICS 2002),2002.
5. D.Boneth and J.Shaw. Collusion secure fingerprint for digital data. IEEE Transactions on Information Theory, 44(5):1897-1905,1998.
6. H. Guth and B. Pfitzmannn. Error and collusion secure fingerprinting for digital data. In Information Hiding '99, LnCS 1768, Springer-Verlag, pages 134-145,2000.
7. R.Sion, M.Atallah, and S. Prabhakar. On watermarking numeric datasets. In Proc. First International Workshop on Digital Watermarking(IWDW 2002), 2002.
8. R.Sion, M.Atallah, and S. Prabhakar. Rights protection for relational data. In Proc. ACM International Coference on Management of Data(SIGMOD 2003), 2002.
9. E.Bertino, S. Jajodia, and P. Samarati. A flexible authorization merchanism for relational data management systems. ACM Transactions on Information Systems, 17(2):101-140, 1999.
10. D. Gross-Amblard. Query-preserving watermarking of relational databases and xml documents. In Proc. of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Pringciples of Database Systems(PODS 2003), 2003.
11. S. Katzenbeisser and E.F. Petitcolas. Information Hiding Techniques for Steganography and Watermarrking. Artech House, Boston MA, 2000.
12. S. Khanna and F.Zane. Watermarking maps: hiding information in structured data. In Proc. Symposium on Discrete Algorithms(SPDA),2000.
13. W. Stallings. Cryptographic Techniques and Data Security. Third Edition. Prentice Hall, 2002.