A New Scheme to Fingerprint XML Data[*]

Fei Guo[1], Jianmin Wang[1], Zhihao Zhang[1], and Deyi Li[1, 2]

[1] School of Software, Tsinghua University, Beijing 100084, China
f-guo03@mails.tsinghua.edu.cn
jimwang@tsinghua.edu.cn
zhangzh02@mails.tsinghua.edu.cn
[2] China Constitute of Electronic System Engineering,
Beijing 100039, China
ziqinli@public2.bta.net.cn

Abstract. Watermarking has been used for digital rights protection over different types of contents on the Web. Since XML data is widely used and distributed over the Internet, watermarking XML data is of great interest. In this paper, we present a new watermarking scheme to embed different fingerprints in XML data. The fingerprint can be used to trace illegal distributors. We also take into consideration that XML data usually contains categorical elements which can't tolerant much modification, our scheme attempts to reduce modifications without bringing down the robustness of the fingerprint. Modifications could be reduced by choosing different patterns to insert. The experiments show that our scheme is effective to make less distortion to the original data and the fingerprint maintains the same robustness level at the same time.

1 Introduction

Today, mass of data could be copied and distributed on the Web easily. Since valuable data could be resold for illegal profit, it's important to claim original ownership of a redistributed copy and trace traitors as well. Watermarking is a class of information hiding techniques to protect digital contents from unauthorized duplication and distribution by introducing small errors into the object being marked. These small errors constitute a watermark or fingerprint. Fingerprinting is often discussed in comparison or extension to watermarking [5]. A watermark is used to identify the owner while a fingerprint is used to identify illegal distributor. For example, the owner embedded a unique fingerprint to each user (user1, user2, user3), see figure 1. When an unauthorized copy on the web is found, the owner could detect the user3's fingerprint to argue ownership and track back to identify user3 to be the illegal distributor out of other users.

Since XML is designed especially for applications on the Web and more and more Internet contents are defined in XML, it's of great significance to fingerprint valuable XML documents.

[*] This research is supported by National Natural Science Foundation of China under Project No. 60473077.

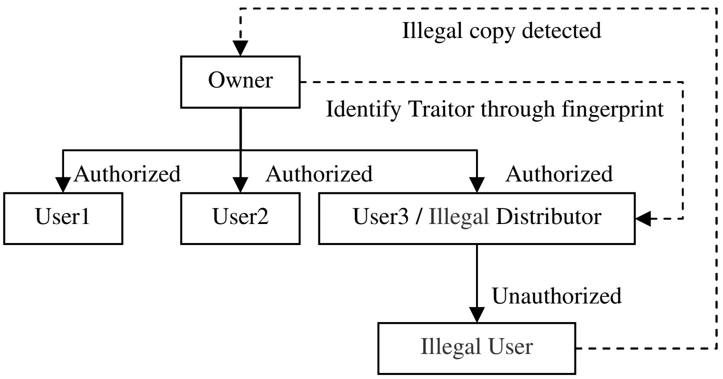

Fig. 1. Using a fingerprint to identify illegal distributors

Wilfred Ng and Ho-Lam Lau [4] present a selective approach of watermarking XML. It's successful to insert a watermark to prove ownership over XML data, but it can't insert different fingerprints to help identify illegal distributors. Sion [2] presents an even grouping method for fingerprinting relational data. We extend his techniques into a varied-size grouping method to fingerprint XML data. Agrawal [1] presents an effective watermarking technique for relational data to modify some bit positions of some attributes of some tuples and gives good mathematic analysis. It gives the foundation of our analysis on confidence level within each group.

In this paper, we introduce a scheme to embed fingerprints of ordered bits. We first classify elements into groups and embed one bit of the fingerprint in each group. To maintain the same grouping result for successful detection, we introduce a varied-size grouping method. A value of "remainder" of each element is calculated to identify which group it belongs to, and the ascending order of the "remainder" naturally preserves the order of the groups, also the order of the fingerprint. Thus, only the number of groups that equals the length of the fingerprint is needed to calculate the same "remainder" when detecting the fingerprint. The even grouping method [2] has to record extra classifying information for each group, which is of the same size of the fingerprint or even more and is not necessary.

All robust watermarking schemes [1] [2] [4] have to make some distortions to the original data. So it's assumed that small errors will not decrease the usability of the original data remarkably in all robust watermarking algorithms. For example, to embed a mark, number <byteCount>5440</byteCount> can be modified to <byteCount>5439</byteCount>, word <TEAM_CITY>Los Angeles</TEAM_CITY> can be replaced by its synonym <TEAM_CITY>L.A.</TEAM_CITY>. But it's hard to define what change is within the acceptable level. In many real life cases, changes tend to be too big to meet the assumption, especially for categorical data. In [7], the distortion to each selected categorical item may be significant, for example, even one bit error in such as social security number is not acceptable. However, [7] assumed that it's acceptable if only a small part of data is modified. So we attempt to find a way to minimum the part to be modified to minimum the change to the original data, meanwhile preserve the same robustness level of the fingerprint. We believe that data with fewer errors is more valuable than data with more errors, both for categorical data and numeric data. In our scheme, we use the fingerprint bit to be embedded to choose the inserting positions, thus different fingerprint bit is represented by different positions, not by the value of the selected mark positions, so we have a choice to either set the selected bit value or word value to "1" or "0", corresponding to either pattern_1 or pattern_0. Since some of the selected mark positions may meet the pattern naturally, i.e., no need to be modified, we can choose a pattern that needs the minimum modifications. In other words, we examine the original values of selected mark positions, and choose a pattern that original values tend to be, thus minimum modifications. So we don't reduce the selected bit positions to minimum modifications, which mean that we don't bring down the mark ratio, thus the fingerprint maintains the same robustness level. Our experiment shows that in some cases, we can reduce the modifications by 1/4 compared with [4] at the same mark ratio. For numeric data, it means less effect on mean and variance; for categorical data, it means we reduce the probability of destroying an element (e.g. any distortion on the social security number) to 3/4.

The rest of the paper is organized as follows: Part 2 provides our insertion algorithm and detection algorithm. Part 3 gives the implementation of our fingerprinting scheme and the analysis on modification amount and fingerprints' robustness. We conclude with a summary and directions for future work in Part 4.

2 Scheme to Fingerprint XML data

In this section, we provide our insertion and detection algorithms. The notations used in this paper are shown in Table 1:

Table 1. Notations

$1/\gamma$	Target fraction of elements marked / mark ratio
ε	Number of candidate bits to be modified
k	The secret key
α	Significance level for detecting a watermark
PA	Primary Data in each element
N	Number of elements in XML data
o	Concatenating

2.1 Insertion Algorithm

A primary data (PA) used to identify each element should be predefined by the owner; also the candidate bit positions and candidate word positions *num_of_word_in_value* should be predefined. The primary data which is used as the primary key in relational databases should be unique and unchanged. For example, the <social_security_number> could be used as PA. If no such data exists, we can construct a virtual PA as stated in [5]. For example, we may use the combination of <SURNAME> and <GIVEN_NAME> instead. We use a one-way hash function value affected by the PA and the secret key k to decide the group position and mark position. Since only the owner knows the secret key, it's hard for an attacker to find our fingerprint.

First we transform a fingerprint in any form (e.g. fingerprint of a picture) into a bit flow and the length of the fingerprint should be recorded for detection. Then we calculate the remainder i for each element at line 4 in our insertion algorithm below. Then elements with same values of i and meet line 5 at the same time are collected into the same i^{th} group. The i^{th} bit of the fingerprint will be inserted into elements in the i^{th} group. Thus we have fpt_length (bit number of the fingerprint) groups. The ascending order of i ranging from 0 to fpt_length-1 naturally preserves the order of the fingerprint. Since the hash result of MD5 we used is expected to be uniform distributed, each group may have varied but similar number of elements.

Let's see how a bit of fingerprint is inserted in each group. We use the fingerprint bit value to choose the inserting positions, see line 5, 13 and 16. It decides which element to mark, and which bit or word to be modified. The mark ratio is used to choose the insertion granularity. Notice that the elements selected to mark and the modification position j are different when the fingerprint to be embedded is "1" from when it's "0". In subroutine pattern($subset_i$), we count the original values of each selected position within a group and choose the mark pattern. Since most categorical data is in textual form, we use the parity of the word's Hash value to represent value "1" or "0" corresponding to bit value for numeric data, see subroutine value(word). For $pattern_1$, we set each selected position value into "1", and for $pattern_0$, we set each selected position value into "0". For example, if the selected values are eight "1" and two "0" in a group, $pattern_1$ is chosen (see line 32) and only two elements have to be changed. In the opposite situation, if the selected values are eight "0" and two "1" in a group, $pattern_0$ is chosen (see line 31) and only two elements have to be changed too. Then subroutine embed($subset_i$) will modify the selected positions according to the pattern chosen, two elements in the example. How to modify the selected position for numeric and textual element is shown at line 14 and 18 respectively.

Algorithm 1. Insertion algorithm

```
// Only the owner of data knows the secret key k.
// R is the XML document to be marked.
// fpt_length is the length of the fingerprint embedded.
1)  fpt[] = bit(fingerprint)
2)  record fpt_length        // length of the fingerprint is recorded for detection
3)  foreach element $\tau \in$R do {
4)    i = Hash(PA o k) mod fpt_length        // fpt[i] to be inserted
5)    if(Hash(fpt[i] o PA o k) mod $\gamma$ equals 0) then   // mark this element
6)      $subset_i$ ← element }
7)  foreach $subset_i$
8)    embed($subset_i$)
9)  subroutine embed($subset_i$)
10)   mask[i] = pattern($subset_i$)
11)   foreach element in $subset_i$ do {
12)     if(element is numeric)
13)       j = Hash(PA o k o fpt[i]) mod $\varepsilon$
14)       set the $j^{th}$ bit to mask[i]        // modify the $j^{th}$ candidate bit
```

```
15)       else if(element is textual)
16)         j = Hash(PA o k o fpt[i]) mod num_of_word_in_value
17)         if(value( the j^th word ) is not equal to mask[i] )     // modify the j^th word
18)           replace the j^th word by a synonym s where value(s) equals mask[i]
19)         else do nothing }
20) subroutine pattern(subset_i)        // choose a pattern for less modification
21)   count_0 = count_1 = 0
22)   foreach element in subset_i do {
23)     if(element is numeric)
24)       j = Hash(PA o k o fpt[i]) mod ε
25)       if(the j^th bit equals 0) count_0 increment
26)       else count_1 increment
27)     else if(element is textual)
28)       j = Hash(PA o k o fpt[i]) mod num_of_word_in_value
29)       if(value( the j^th word )) count_0 increment
30)       else count_1 increment }
31)   if(count_0 > count_1) mask = 0     // pattern_0
32)   else mask = 1                      // pattern_1
33)   return mask
34) subroutine value(word)
35)   if(Hash( word ) is even)
36)     value = 0
37)   else value = 1
38)   return value
```

2.2 Detection Algorithm

To detect a fingerprint, the owner has to use the same secret key, the same predefined parameters, the fingerprint length recorded when inserting and choose a significance level for detection.

First we form similar groups, see line 3 in our detection algorithm below, thus preserve the same fingerprint order. Next we try to detect one bit of fingerprint from each group. If the embedded fingerprint is “0”, compared with the insertion process, we can find exactly the same elements at line 10, and the same selected positions at line 13 and 17. For a non-marked document, because the positions are selected randomly, the probabilities of a selected position value to be either “0” or “1” are both 1/2 approximately. But for a marked document, we are expected to see that the values of each selected position are all the same, either “0” or “1”, i.e., $match\_count_0$ = $total\_count_0$ or $match\_count_0$ = 0. We use the significance level set by the owner to calculate a threshold (see line 35), such that either if $match\_count_0$ is larger than threshold or is smaller than ($total\_count_0$ – threshold), we can claim that a fingerprint bit of “0” has been embedded with the confidence level of (1 -), otherwise, we say a fingerprint bit of “0” isn’t detected. Then we check if the embedded fingerprint is “1” (see line 20), the process is almost the same. If both “1” and “0” haven’t be detected, we conclude that no fingerprint has been embedded at the confidence level of (1 - α).

Algorithm 2. Detection algorithm

```
// $k$, $\gamma$, $\varepsilon$ and num_of_word_in_value have the same values as in watermark insertion.
// fpt_length has the same value with recorded when inserting.
// $\alpha$ is the significance level for detecting a fingerprint bit.
// S is a marked XML document.
1)  foreach element $\tau \in S$ do {
2)    i = Hash(PA o $k$) mod fpt_length
3)    subset$_i$ ← element }
4)  foreach subset$_i$
5)    detect(subset$_i$)
6)  return fpt[]

7)  subroutine detect(subset$_i$)      // recover one bit from each subset
8)    total_count$_0$ = match_count$_0$ = total_count$_1$ = match_count$_1$ = 0
9)    foreach element in subset$_i$ do
10)     if(Hash(0 o PA o $k$) mod $\gamma$ equals 0) then {      // subset_0
11)       total_count$_0$ increment
12)       if(element is numeric)
13)         j = Hash(PA o $k$ o 0) mod $\varepsilon$
14)         if(the j th bit equals 0) then
15)            match_count$_0$  increment
16)       else if(element is textual)
17)         j = Hash(PA o $k$ o 0) mod num_of_word_in_value
18)         if(Hash( the j th word) is even) then
19)            match_count$_0$  increment }
20)     if(Hash(1 o PA o $k$) mod $\gamma$ equals 0) then {      // subset_1
21)       total_count$_1$  increment
22)       if(element is numeric)
23)         j = Hash(PA o $k$ o 1) mod $\varepsilon$
24)         if(the j th bit equals 0)
25)            match_count$_1$  increment
26)       else if(element is textual)
27)         j = Hash(PA o $k$ o 1) mod num_of_word_in_value
28)         if(Hash( the j th word) is even)
29)            match_count$_1$  increment }
30)   if(match_count$_0$ > threshold(total_count$_0$, $\alpha$)) or
      (match_count$_0$ < total_count$_0$ - threshold(total_count$_0$, $\alpha$))   // pattern$_0$?
31)      return fpt[i] = 0
32)   else if(match_count$_1$ > threshold(total_count$_1$, $\alpha$)) or
      (match_count$_1$ < total_count$_1$ - threshold(total_count$_1$, $\alpha$))   // pattern$_1$?
33)      return fpt[i] = 1
34)   else return False             // no pattern
35) subroutine threshold(n, $\alpha$)
36)   return minimum integer m that satisfies $\sum_{k=m}^{n} C_n^k \left(\frac{1}{2}\right)^n < \frac{\alpha}{2}$
```

The selection process in our detection algorithm can be modeled as a Bernoulli trial, thus the match_count in a non-marked document is a random variable that meets a binominal distribution with parameters total_count and 1/2. Thus the threshold should satisfy (1) below.

$$P\{\text{MATCH\_COUNT} > \text{threshold} \mid \text{total\_count}\} + P\{\text{MATCH\_COUNT} < \text{total\_count} - \text{threshold} \mid \text{total\_count}\} < \alpha \quad (1)$$

Based on Agrawal's mathematic analysis [1], the threshold for a given total_count at confidence level of 1 - α can be calculated using formula (2) shown below..

$$\text{threshold} = \text{minimum integer m that satisfies } \sum_{k=m}^{\text{total\_count}} C_{total\_count}^{k} \left(\frac{1}{2}\right)^{\text{total\_count}} < \frac{\alpha}{2} \quad (2)$$

Thresholds for total_count from 1 to 30 when α = 0.01 are shown in figure 2 below. We can see that the bigger total_count is, the smaller portion of threshold is. Thus, given a large total_count, it gives the potential to resist attacks. For example, when total_count is 100, the threshold is only 64, which means with loss of nearly 40% of the mark, the fingerprint bit will still be detected successfully at the confidence level of (1 - α).

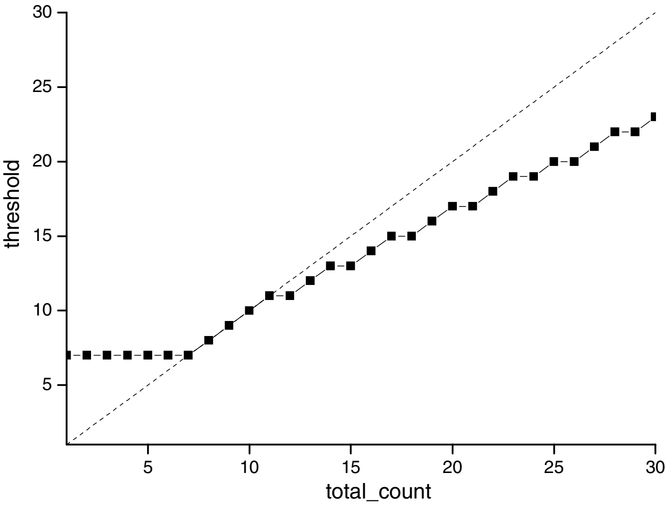

Fig. 2. The relationship between total_count and threshold when α = 0.01

3 Experiments and Analysis

We ran experiments in Windows 2003 with 2.0 GHz CPU and 512MB RAM. The XML data source is weblog.xml. For simplicity, we choose numeric elements to modify, results for textual elements are almost the same. We set = 10, = 3 and = 0.01, insert a 100-bit long fingerprint which can identify 2^{100} different distributors. We choose N_1 = 100,000 and N_2 = 10,000 of the records and experiment separately.

First we see our varied-size grouping method in figure 3, we list 10 groups. It shows that the total selected elements are almost *1/* of N and each group has varied but similar sizes. It means no element or few elements are selected in a certain group seldom happens. Some marks are expected in each group, thus we can have an entire fingerprint.

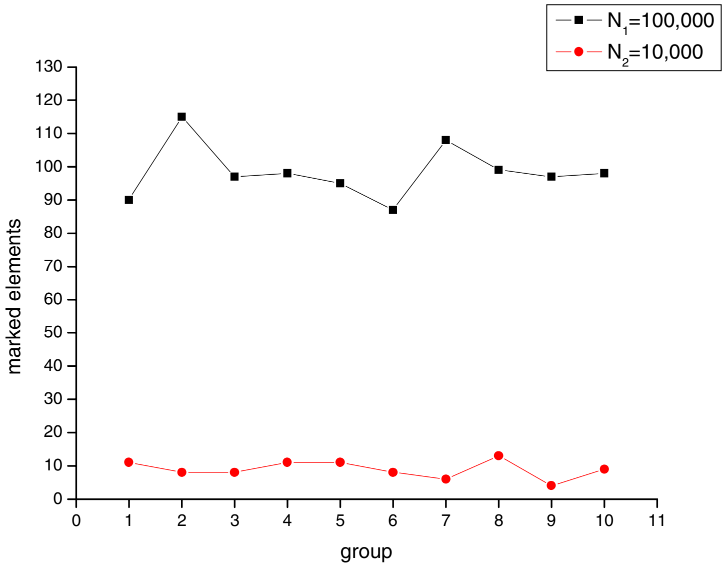

Fig. 3. Elements selected in each group

The situations in each group are alike with Wilfred Ng's selective approach [4]. So we can compare the amount of modifications in our scheme with Wilfred Ng's approach. We use the same secret key and the same other parameters to embed the same fingerprints. We can see in table 2 that the selected elements are all the same. Also the grouping results are the same. Because all parameters used in insertion are the same. When N_1 = 100,000, the elements needed to be modified are 4642 out of 9995 selected positions in our method. Compared with Wilfred Ng's method, it's 4960 elements to be modified out of 9995 selected positions. We reduce modify-cations by 6.4%. When N_2 = 10,000, the elements needed to be modified are

Table 2. The amount of modifications compared with Wilfred Ng's selective approach

	Modifications (our method)	Selected elements (our method)	Modifications (Ng's)	Selected elements (Ng's)	
N_1 = 100,000	4642	9995	4960	9995	93.6%
N_2 = 10,000	374	984	491	984	76.1%

374 out of 984 selected positions in our method. Compared with Wilfred Ng's method, it's 491 elements to be modified out of 984 selected positions. We reduce modifications by 23.9%.

We can see a significant reduction of modifications when N_2 = 10,000, not too significant when N_1 = 100,000. The reason is that when N_1 = 100,000, each group has about 100 elements; when N_2 = 10,000, each group has about 10 elements. It's nearer to 50% to be modified when N is bigger. It's like throw a coin. For example, if you throw 10 times, the number of times when head is up is fluctuating around 5 heavily. If increased to 100 times, the number of times when head is up will be near 50. So the bigger N is, the reduction of modifications is less significant, see figure 4, we show 10 groups.

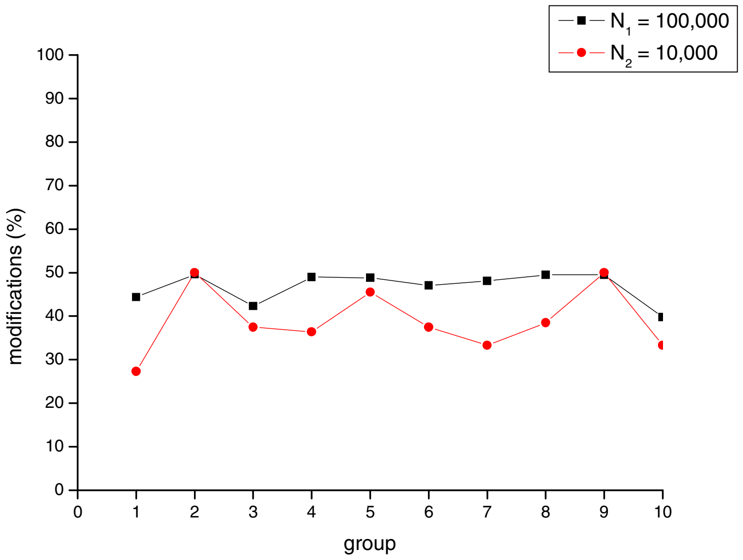

Fig. 4. Portion of elements needed to be modified

We can look at each group as an XML document input in Wilfred Ng's selective approach, thus we can compare the robustness level of our fingerprint bit in each group with Ng's approach. Because the confidence level is decided by the selected positions, so the robustness level of our fingerprint bit in each group is the same with Ng's result. We can see in table 2 that although the modifications are reduced by 23.9% when N_2 = 10,000, the number of selected positions are both 984 elements, so the robustness level maintains the same.

4 Summary

In this paper, we present our new watermarking scheme to embed fingerprints in XML data. Thus, we can not only prove ownership, but also identify illegal distributors since a unique fingerprint is embedded in each copy delivered to different distributors. We use a varied-size grouping method to preserve the order of the fingerprint's bits. To solve the problem of some categorical elements in XML document can't tolerant much modification, we make our effort to reduce modifications at the same insertion level, i.e., without bringing down the robustness of the fingerprint. In our scheme, to minimum modifications, we use the fingerprint to decide the inserting positions and then choose an inserting pattern. The experiments show that our scheme is effective to make less distortion to the original data and the fingerprint maintains the same robustness level at the same time. In some cases, we can reduce the modifications by 1/4.

In the future, we would like to research on the confidence level of the whole fingerprint, especially when part of the fingerprint has been destroyed; and how to argue ownership and identify illegal distributors from a fragmentary fingerprint.

References

1. Rakesh Agrawal, Peter J. Haas, Jerry Kiernan.: Watermarking Relational Data: Framework, Algorithms and Analysis. VLDB Journal. (2003).
2. Radu Sion, Mikhail Atallah, Sunil Prabhakar.: Rights Protection for Relational Data. Proceedings of ACM SIGMOD. (2003) 98–109.
3. David Gross-Amblard.: Query-preserving Watermarking of Relational Databases and XML Documents, PODS 2003, San Diego CA. (2003)191–201.
4. Wilfred Ng and Ho-Lam Lau.: Effective Approaches for Watermarking XML Data. DASFAA 2005, LNCS 3453, pp. 68–80, 2005.
5. Yingjiu Li, Vipin Swarup, Sushil Jajodia.: Constructing a Virtual Primary Key for Fingerprinting Relational Data. DRM'03, Washington, DC, USA. 2003.
6. Radu Sion, Mikhail Atallah, Sunil Prabhakar.: Resilient Information Hiding for Abstract Semi-Structures. Proceedings of IWDW. 2004.
7. Radu Sion, Proving Ownership Over Categorical Data. Proceedings of ICDE 2004, 2004.
8. Yingjiu Li, Huiping Guo, Sushil Jajodia.: Tamper Detection and Localization for Categorical Data Using Fragile Watermarks. DRM'04, Washington, DC, USA. 2004.