Effective Approaches for Watermarking XML Data

Wilfred Ng and Ho-Lam Lau

Department of Computer Science,
The Hong Kong University of Science and Technology, Hong Kong
{wilfred, lauhl}@cs.ust.hk

Abstract. Watermarking enables provable rights over content, which has been successfully applied in multimedia applications. However, it is not trivial to apply the known effective watermarking schemes to XML data, since noisy data may not be acceptable due to its structures and node extents. In this paper, we present two different watermarking schemes on XML data: the selective approach and the compression approach. The former allows us to embed non-destructive hidden information content over XML data. The latter takes verbosity and the need in updating XML data in real life into account. We conduct experiments on the efficiency and robustness of both approaches against different forms of attack, which shows that our proposed watermarking schemes are reasonably efficient and effective.

1 Introduction

Watermarking in the contexts of image, audio or video data is well-known to be an effective technique to protect the intellectual property of electronic content. Essentially, the technique embeds a secret message into a cover message within the content in order to prove the ownership of materials. Remarkable successes in watermarking on multimedia applications have been achieved in recent years [4]. Thus, relevant business sectors are able to distribute their data while keeping the ownership and preventing the original data being resold illegally by others.

The existing watermarking technology has mostly been developed in the context of multimedia data, since such data has a high tolerance to noise and thus it is not easy to detect the watermark. Unlike multimedia data, XML data are diverse in nature: some are data-centric and numeric (e.g. regular scientific data) while some are document-centric and verbose (e.g.book chapters). It is challenging to develop an effective watermarking scheme which is invisible and is able to resist various kinds of attack.

In this paper, we attempt to develop watermarking schemes for XML data based on two different watermarking approaches. One is the *selective approach* and another is the *compression approach*. As for the selective approach, we develop a watermarking scheme for uncompressed XML data based on the database watermarking algorithm proposed by Agrawal [2]. The second approach is more interesting. It follows our advocation that in reality some XML documents are verbose and they need compression in practical applications [3]. In addition, we

L. Zhou, B.C. Ooi, and X. Meng (Eds.): DASFAA 2005, LNCS 3453, pp. 68–80, 2005.

take into consideration that XML documents need to be updated frequently. Therefore, in the compression approach, we introduce a novel watermarking scheme based on our earlier developed XML compressor, namely XQzip, which does not require full decompression when querying. [3]. By watermarking compressed XML data, we gain the advantage of having better document security, and at the same time, higher flexibility of updating XML data.

Related Work. Agrawal presents an effective watermarking technique for the relational data [2]. This technique ensures some bit positions of certain attributes contain the watermarks. We extend their techniques on XML data by defining locators in XML in our selective approach. Sion [5] discusses the watermarking of semi-structures of multiple types of contents and represents them as graphs by characterizing the values in the structure and individual nodes. He also proposes a watermarking algorithm that makes use of the encoding capacity of different types of nodes. Gross-Amblard [1] investigates the problem of watermarking XML databases while preserving a set of parametric queries. His work mainly focuses on performing queries on different structures and pay less attention to the watermarking scheme. However, the query approaches are similar to the predefined queries used in the compression approach. At present, all proposed XML watermarking schemes are based on uncompressed XML data and no studies exist on watermarking compressed XML data to the best of our knowledge.

Paper Outline. After introducing our XML watermarking schemes in this section, we describe and study the selective approach, which is for uncompressed XML data, and the compression approach, which is for XQzip compressed XML data in Sections 2 and 3, respectively. Then in Section 4, we conclude our work and suggest future improvements for the watermarking schemes we developed.

2 The Selective Approach of Watermarking XML

In this section, we introduce the selective approach of XML watermarking. We also analyze the robustness of our watermarking system against the following two forms of attacks: *subtractive attack* and *additive attack*. All the experiments related to the watermarking system are conducted on a machine of the configuration as follows: P4 2.26GHz, 512MB main memory and 15GB disk space.

2.1 Watermark Insertion

In the selective approach of watermarking XML, the watermarks are randomly distributed throughout the XML document based on a secret key provided by the owner. We aim at making minor changes on XML data without causing errors during the process.

The watermark insertion algorithm and the notations we used are presented in Algorithm 1 and Table 1, respectively. Before embedding marks in XML data, we define a *locator*, which is an analogy to the primary key in relational databases, to indicate whether a particular element should be marked. Unlike

watermarking relational databases [2], primary keys are not necessarily to be specified and defined in XML. We assume the owner of the watermarked data is responsible to select the elements that are suitable to be the candidates of locators. The best choice of such an element is that its value is unique, non-modifiable and has large locator space. For example, the tag "ISBN" of a book XML document could be served as a locator.

Table 1. Notation Used

υ	Number of elements in the document available for marking.
ξ	Number of the least significant bits available for marking in an element.
$1/\gamma$	Fraction of marked elements in the document (the watermark ratio).
α	Significance level of the test for detecting a watermark.
N	Number of elements in the document.
τ	Minimum number of detected locators required for a successful detection.

Let E be a locator candidate and K be the secret key provided by the owner. We use the value of E in a hash function $H(E, K) = E \circ K$, which generates a hash value, h to determine whether E should be marked or not. For the sake of simplicity, we use concatenation to generate h. In fact $H(E, K)$ can be some other functions as long as it is able to generate a unique value for a given pair of E and K. After determining the marked locators, say E_m, we watermark the value of E_m according to the data type of E_m. For numerical data, we mark E_m by modifying the least significant bit specified by the control parameter, denoted as ξ. We assume that numerical data is able to tolerate small and non-detectable changes. For example, "1000.30000" can be changed to "1000.30001". For textual data, the value of E_m is replaced by a synonym function, denoted as $Synm()$, which is based on a well-known synonym database WordNet [10]. For example, "theory" can be replaced by its synonyms "concept" or "belief". Once E_m is marked, we call it the *marked element*.

2.2 Watermark Detection

To detect whether the XML data has originated from the data source, the data owner is required to supply the secret key, K, and the corresponding *setting file* to the watermark detection algorithm, which is shown in Algorithm 2. The setting file includes information such as the significance level, α, and the list of locator candidates in XPath format. The detection algorithm finds out the number of marked elements and locators in the XML data, and then evaluates the *hit rate*. We call the locator whose watermark is detected by the algorithm the *detected locator*. The detection algorithm uses a threshold function to calculate the smallest integer, denoted as t, such that if the hit rate is larger than t, the owner can claim the ownership of the document with the confidence of $(1 - \alpha)$.

Figure 1 shows the proportion of detected locators required for a successful detection with 99% confidence against different watermark ratios. It is interest-

Algorithm 1 The watermark insertion algorithm in the selective approach

```
1: for each locator candidates r ∈ R do{
2:   if (r.lablel() mod υ equals 0){ //mark this locator
3:     value_index i = r.lablel() mod υ; //modify value A_i
4:     if (A_i is textual){
5:       word_index w_i = r.lablel() mod num_of_word_in_value;
6:       A_i = markText(r.lablel(), A_i, w_i);} //modify the w_i^th word
7:     else if (A_i is numerical){
8:       bit_index b_i = r.lablel() mod ξ; //modify the b_i^th bit
9:       A_i = markNum(r.lablel(), A_i, b_i);}}}
10: Procedure markNum(secret_key sk, number υ, bit_index j) return number
11:   first_hash = H(K · sk)
12:   if(first_hash is even)
13:     set the j^th least significant bit of υ to 0;
14:   else
15:     set the j^th least significant bit of υ to 1;
16:   return υ;
17: Procedure markText(secret_key sk, text υ, word_index j) return text
18:   first_hash = H(K · sk);
19:   if (first_hash is even)
20:     replace the j^th word w by a synonym s where s = change(w,0);
21:   else
22:     replace the j^th word w by a synonym s where s = change(w, 1);
23:   return υ;
24: Procedure change(word w, value υ) return word
25:   if (Symn(w) equals υ)
26:     Do nothing and return w;
27:   else{
28:     syn_list = all synonyms of w from a dictionary database;
29:     randomly select a synonym s from syn_list where Synm(s) equals υ;}
30: Procedure Synm(word w) return number
31:   if (H(w) is even)
32:     return 0;
33:   else
34:     return 1;
```

ing to find that, in a small XML document (N = 10,000), if 1% of the records are marked, only 62% of detected locators are needed to provide 99% confidence. As the watermark ratio increases, the proportion of detected locators required decreases. The proportion tends to the constant value of 0.5 because the detection algorithm is probabilistic and needs more than 50% of detected locators to differentiate a watermark from a chance of random occurrence. In general, for an XML document with more elements, fewer detected locators are required to achieve the same level of detectability than XML documents with fewer elements.

Algorithm 2 The watermark deletion algorithm in the selective approach

```
1: TotalCount = 0;
2: MatchCount = 0;
3: for each locator r ϵ R do{
4:   if (r.lablel() mod γ equals 0){// this locator is detected
5:     value_index i = r.lablel() mod υ; // value A_i was modified
6:   if (A_i is textual) {
7:     word_index w_i = r.lablel() mod num_of_word_in_value; // w_i^th word was modified
8:       TotalCount = TotalCount+1;
9:       MatchCount = MatchCount + isMatchNum(r.lablel(), A_i, w_i);}
10:   else if (A_i is numeric) {
11:      bit_index b_i = r.lablel() mod ξ // b_i^th bit was modified;
12:      TotalCount = TotalCount+1;
13:      MatchCount = MatchCount + isMatchWord(r.lablel(), A_i, b_i^th);}}}
14: t = Threshold(Totalcount, α, 0.5 );
15: if(MatchCount ≥ t)
16:    The document is a suspect piracy;
17: Procedure Threshold(number n, significance a, success probability p)
18:    q = 1 − p;
19:    return minimum integer k such that Σ_{r=k}^{n} nCr p^r q^{n−r} < α;
```

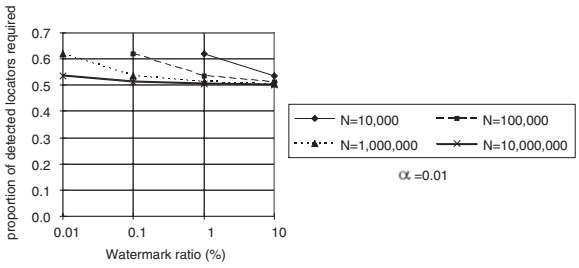

Fig. 1. Proportion of detected locators required for successful watermark detection

2.3 Experiments

Two XML data sources are used in the experiments: *1998_statistics.xml* and *weblog.xml*. Table 2 shows the features of these datasets.

Marked and Modified Records. We first examine the relationship between the fraction parameter, γ, and the marked locators. Figure 2(a) shows that the percentages of marked locators in the two XML datasets are slightly lower than our expected level (c.f. the superimposed curve representing $1/\gamma$). This is due to the fact that some locators cannot be modified by the watermark insertion algorithm, such as the synonym of a word of the locator does not exist. Figure 2(b) shows that the percentage of modified locators with different values of γ. The experimental result for "1998_statistics.xml" is fluctuating around our expected level of 50%, while that of the "weblog.xml" is usually less than 50%.

Table 2. Features of XML datasets

Documents	File size (KB)	No. of records	No. of elements available for marking (υ)	No. of least significant bits for marking (ξ)
1998_statistics.xml	1227	1226	4	3
weblog.xml	89809	247024	2	3

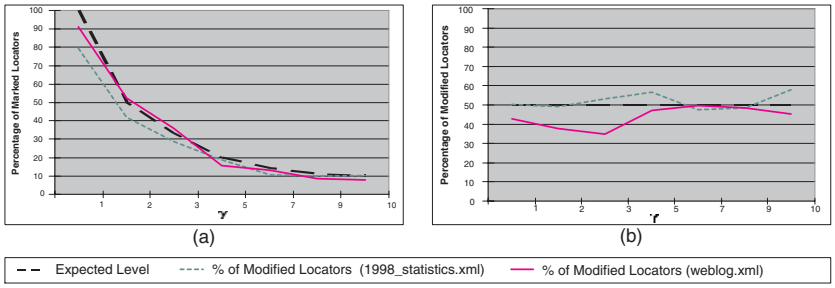

Fig. 2. Percentages of (a) marked and (b) modified locators versus γ

Table 3. Running time of watermark insertion

Watermark ratio (%)	100.00	50.00	33.33	20.00	14.29	11.11	10.00
1998_statistics.xml (sec)	0.891	0.891	0.861	0.891	0.871	0.871	0.851
weblog.xml (sec)	73.46	69.02	68.01	61.68	61.62	61.44	61.79

Running Time of Watermark Insertion. Table 3 shows the running time of applying the watermark insertion algorithm on the two XML datasets with different watermark ratios. The results show that the watermark ratios do not have a big impact on the running time of the algorithm. The I/O time is the main overhead, since the document is parsed only once, which is irrespective to the watermark ratios.

Subtractive Attack. A *subtractive attack* aims at eliminating the presence of watermarks. A successful subtractive attack reduces the watermarks created by the original owner in order to render the claim of ownership impossible. *Subset attack* is a typical form of subtractive attacks, it attempts to copy parts of the watermarked document and hence reduces the percentage of watermarks found in the document. We use the "weblog.xml" dataset to demonstrate the resistance to subset attacks in the selective approach. We randomly select elements from the watermark version of weblog.xml at different gap sizes and selectivity levels and then examine the watermark detection percentage.

In Figure 3, when the gap size is equal to 10, 90% of watermarks can be detected with only 0.02% selectivity level. When gap size increases, selectivity level also increases for detecting over 90% of watermarks. At 0.3% selectivity, watermark detection reaches 100%. For a gap size of 1000, only a small selectivity level can reach over 90% watermark detection. This result indicates that the watermarks inserted by our watermarking schemes are evenly distributed and have good resistance to the subset attack.

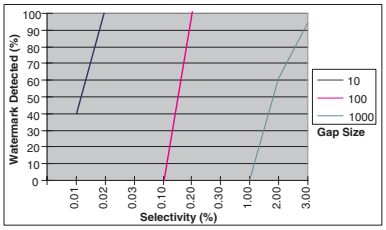

Fig. 3. Watermark detected versus selectivity level

Additive Attack. In an additive attack, illegal parties insert their own watermarks over the original document and claim the "legal" ownership of the data. Since the watermarks inserted afterwards is able to overwrite the former watermarks in some overlapping regions, it results in the illegal copy more detected elements than the original one can be found in the overlapping regions. Let M be the total number of marked element, L be the number of elements available for marking and F be the total number of watermarks added afterwards. The probability of having overlapping region is given as follows:

$$\begin{cases} 1 - \Pi_{i=0}^{F-1} \frac{(L-i)-M}{L-i}, & \text{if } M+F < L; \\ 0, & \text{if } M+F \geq L. \end{cases}$$

The mean of the overlapping region $= L \times$ Probability of collision in an element node $= L \times (M/L) \times (F/L)$.

Illegal parties may try to reduce the overlapping regions by using a low watermark ratio such as 0.1% or 0.01%. Figure 4 shows the probability and the mean of having overlapping regions when the watermark insertion algorithm is applied twice on the same XML document.

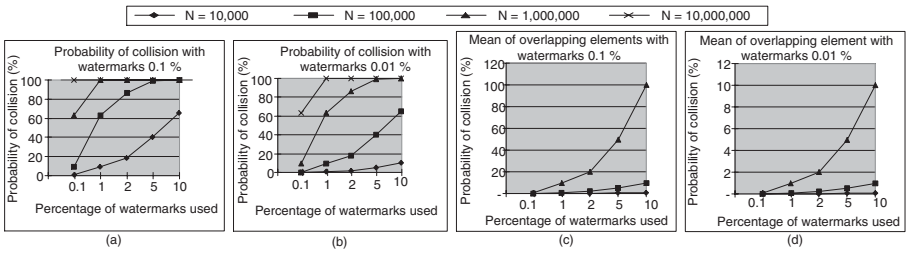

Fig. 4. Probability and mean of having overlapping region with 0.1% and 0.01% watermark ratios

Figure 4(a) shows that for a small XML document (N=10000), if the owner uses a 10% watermark ratio and the illegal party inserts watermarks with a 0.1% watermark ratio, the probability of the occurrence of overlapping regions is 65%. However, the mean of these overlapping regions is only 1 as shown in

Figure 4(c). For a large XML document (N = 100,000), if the owner uses a lower watermark ratio of 2% and the illegal party inserts watermarks with a 0.1% watermark ratio, we can achieve a higher probability of overlapping (85%) and the mean of overlapping region is 2. Figures 4(b) & (d) show the results of illegal parties using a very low watermark ratio of 0.01%. They show that the probability of an overlapping occurrence and the mean of overlapping region decrease dramatically. In this case, since the probability of overlapping region is low, in order to resist additive attacks with a very low watermark ratio, we can decrease the value of γ such that when overlapping occurs, the collisions of watermarks are large enough to make an accurate decision.

Discussions. The experimental results show that the selective approach is susceptible to subtractive and additive attacks. The performance of the approach is determined by four parameters: the size of the document (N), the watermarking ratio (γ), the number of locators (v) and the significant level of the test (α). It is worth mentioning in our finding larger documents can use a smaller watermark ratio to achieve a particular confidence of detectability (c.f. Figure 1). Computation overhead introduced by the watermark ratio to the watermark insertion algorithm is relatively small compared to the I/O time. The overhead is also directly related to the size of the documents.

Our watermark algorithm can also resists some attacks that transform the structure of the watermarked XML data conforming to DTDs or XML schemas (i.e. distortive attack). Since XML data is a tree structure, re-transformation to the original structure is possible by using the original schema, in this case we can still apply the watermark detection algorithm to examine whether the XML data belongs to the owner. In real life, some XML document is based on some well-known schema, such as the Electronic Business using eXtensible Markup Language (ebXML) [8], the distorted documents become less valuable.

3 The Compression Approach of Watermarking XML

Existing watermarking techniques are all targeted on plain XML data. The selective approach we introduced in Section 2 is efficient and effective in proving the ownerships of the owners. However, the protection on the data from access in this approach is not taken into consideration. In this section, we introduce a novel watermark approach which is based on compressed XML data which provides a prove of ownership as well as the protection of data security. The principle is that compressed XML data are unable to be retrieved directly without the correct decompression. Watermarking compressed XML data is a preventive measure for unauthorized copy or reading. Similar to the selective approach, the compression approach is also driven by the owner-selected secret key. The secret key is used to calculate the location of the watermarks, each distributed copy is watermarked by a unique secret key set by the owner. Without the secret key, the compressed data cannot be retrieved. Authorized parties are allowed to retrieve the data by using the pre-defined queries sets provided by the owner, the owner

can also limit the amount of data visible to different parties by giving them different pre-defined query sets. We also consider in practice some XML data are updated frequently and thus it is inefficient to compress and watermark the XML documents again for every update activity. Therefore, we provide a facility which only requires the owner to distribute supplementary compressed files to the relevant parties when update occurs.

3.1 Architecture of the Compression System

The architecture of the compression system is shown in Figure 5. We only briefly explain the functionality of the main modules due to the space limit.

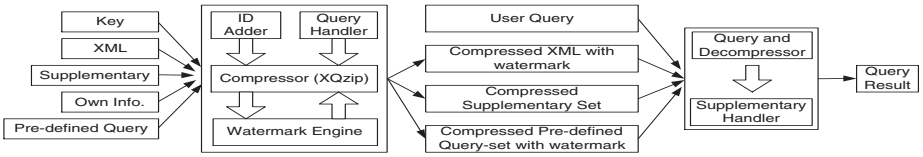

Fig. 5. An overview of the compression approach of watermarking XML

ID Adder. The ID Adder is built on a SAX parser, it parses the XML document sequentially and inserts the owner's information into the XML document. The information is stored in an "ownership node" under the root of the XML document. The label of the ownership node is simply the hash value of the owner's secret key. To support update, a unique system-assigned ID value is added for each element for easy processing when consulting the supplementary file.

Query Handler. This module is an interface that allows the owner to restrict some pre-defined queries for an authorized party. The module selects the visible parts of the XML document, then converts them into an XML document and finally passes the XML document to the compressor.

Compressor and Watermark Engine. We adopt our earlier developed XQzip [3] to carry out the XML compression. The secret key, hash function and gap value are used to determine the byte position to be marked. Roughly, a smaller gap value results in more watermarks being inserted into the compressed blocks. After all the blocks are compressed and watermarked, they are merged into a single file.

Query and Decompressor. Querying and decompressing are also executed using XQzip [3]. A compressed pre-defined queries set is first decompressed. Then, the authorized parties select and perform queries from the pre-defined query list. The system locates the position of the query solution through the index file developed in XQzip. The query solution is decompressed and passed to the supplementary handler if update is required. In the process of decompression,

the query and decompressor hashes the embedded secret keys and gap values to determine the marked elements and recover them.

Supplementary Handler. There are two steps in handling the supplementary files when updating the compressed XML data. First, the handler removes the out-dated contents defined by the supplementary set. The result is then passed to the ID adder which updates the contents defined by the supplementary document. Then, the parser checks for every attributes of each element, and if an attribute is indicated as "added", the parser inserts the value to the corresponding elements and finally, the result is outputted as an XML file.

In the compressed watermark system, the number of watermarks in each compressed block is restricted. The system processes byte flipping at the locator indicated by the hash function. If one byte location is selected twice or even number of times, flipping does not occur at that byte location. To ensure that a block contains at least one watermark, the number of watermark in each compression block should be odd. The number of marks is determined by the following formula where m is the block size: $Number\ of\ mark = \begin{cases} m, & \text{when } m \text{ is odd;} \\ m+1, & \text{otherwise.} \end{cases}$

3.2 Experiments

We implement the compression watermarking system and conduct a series of experiments which are based on the same machine configuration as stated in Section 2. Four common XML datasets are used in the experiments: XMark, Shakespeare and two DBLP data sources of different size.

Effectiveness and Detectability of Compressed Watermarking System. The data of a compressed document is retrieved by using the same {key, gap} pair used in the compression. To test for the effectiveness of the system, we retrieve data from a compressed document by the query system from the following scenarios: (1) different secret keys and different gap values, (2) the same secret key but different gap values, (3) different secret keys but the same gap value, and (4) the same secret key and same gap value. Note that when a wrong {key, gap} pair is supplied, the query and decompressor cannot locate the pattern of marked element and fails to decompress the required data.

Query Response Time. We present the worst case query response time of the four datasets in the system on three different scenrios: (1) ID and update are supported, (2) ID is supported but no update and (3) ID and update are not supported. The test query we used is set to retrieve the whole document and the processing involves a full decompression.

Figure 6(a) shows the query response time of the data sources in the system which support ID and update. When the smallest gap value, i.e. gap = 1, requires roughly 30% more time to process the query than that with the largest gap value, i.e. gap = 10000. The reason for this is that the gap value controls the number of marks to be recovered, the smaller the gap value, the more the marked elements are needed to be recovered. Figure 6(b) shows the query response time of our system and XQzip which support ID but not update. It shows that when the

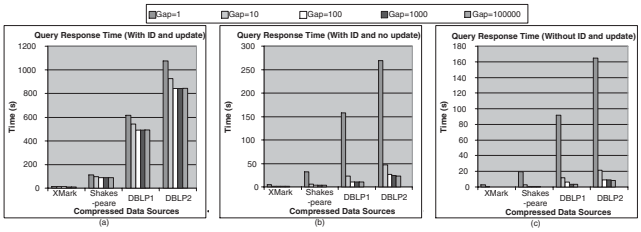

Fig. 6. Query processing time comparison (a) (support ID and update), (b) (support ID but not update) and (c) (do no support ID nor update)

gap value is smaller, the query processing time is longer; it takes 10 times longer for gap = 1 than gap = 10000.

Compared Figure 6(a) to Figure 6(b), there is a big difference in the query time between the updated case and that of not updated. The difference is obvious when the gap value is small and the file size is large, in this case handling supplementary XML documents is expensive and consumes too much time. However, when the gap value is very small, time spent on recovering the marked elements from the compressed XML data becomes critical and the time spent on handling supplementary XML document becomes less significant.

Figure 6(c) shows the query time of the system which support neither ID nor update. The result is similar to the results shown in Figure 6(b). In this case most of the time is used in recovering the marked elements from the compressed XML data. This also indicates that the time required for handling ID is linear to the size of the document, which does not introduce much overhead.

Robustness of Watermarking. We now analyze the robustness of the compression approach against various forms of attacks. An attack is assumed to be aimed at retrieving data from the compressed document without using the query system. Such attacks are classified as a *flipping attack* or an *averaging attack*.

Flipping Attack. A flipping attack attempts to destroy the watermark by flipping the value at certain byte positions. Since a compressed document is composed of many compressed blocks, an attacker attempts to attack each block and combine all the attacked blocks to form a new compressed document. If any one block is modified wrongly, the compressed document is unable to be decompressed therefore the attacker has to guess the pattern of all blocks correctly.

Suppose the attacker knows the number of marked locators, m, and the size of the data, n. If two marked elements are at the same location, they cancel the effect of each other. The probability of correctly guessing the pattern of marks is given by: $\frac{1}{\Sigma_{r=1,3,\ldots,m} nCr}$, for $0 < m \leq n$ and m is odd.

Figure 7 shows the success rate of the simulated flipping attacks. The success rate decreases upon the increase of block size at a fixed gap value. On the other hand, if the block size is fixed, a decrease in gap value decreases the success rate. For instance, a small block size ($n = 30$) and a large gap value (gap = 100000) gives a higher success rate, however, most of the success rates are within 3%, which are rather low.

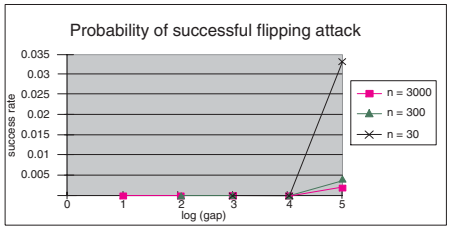

Fig. 7. Success rate for flipping attack

Averaging Attack. An averaging attack attempts to construct a watermark-free document from a number of sample watermarked documents. Similar to the flipping attacker, an averaging attacker attacks the compressed data block by block by analyzing all the compressed blocks from the available samples and uses the information to construct a new block. Let the attacker has s samples (s is supposed to be an odd number) and create a new artificial block, a. The i^{th} byte of a is the majority value of the i^{th} byte in all the s samples.

We carry out experiments by simulating the averaging attack by using the XMark and Shakespeare datasets. For each dataset, there are 99 samples available. We randomly generate the blocks sizes as the attack targets. To launch a successful averaging attack, all artificial blocks are required to be successfully attacked. The number of samples required is shown in Table 4 and "Fails" means the attack is fail after averaging all the samples.

Table 4. Number of samples required for averaging attack at different gap size

Gap Size	1	5	10	100	1000	10000	100000
XMark	Fails	Fails	Fails	5	3	3	3
Shakespeare	Fails	Fails	Fails	5	3	3	3

From Table 4, we find that the attack on compressed data can be easily successful if the gap values are large. This is mainly because majority of blocks being unmarked when the gap value is large. When the gap value is below 10, all attacks fail, since few byte errors are sufficient to prevent the attacker from decompressing the data. From the results, we can see that when the gap value is smaller the system is more robust against the averaging attack. Thus, there is indeed a tradeoff between the gap value and compression time.

4 Concluding Remarks

We have presented two approaches for watermarking XML for both usual XML and a compressed form, which are shown to be robust and effective. The proposed watermark algorithms are presented and various forms of attacks are studied. In the selective approach, we decide how to insert synonyms and control the

synonymity level of a word. The performance of the scheme in this approach depends very much on the quality of the synonym database. In the compression approach, we rely on an effective queriable XML compressor we developed. This approach is both effective and practical for large XML datasets.

References

1. D. Gross-Amblard. *Query-preserving Watermarking of Relational Databases and XML Documents.* In Proc. of Principle of Database Systems, 2003.
2. R. Agrawal and J. Kiernan. *Watermarking Relational Databases.* In Proc. of VLDB, 2002.
3. J. Cheng and W. Ng. *XQzip: Querying Compressed XML Using Structural Indexing.* In Proc. of the EDBT, pages 219-236, 2004.
4. C. Collberg and C. Thomborson. *Software Watermarking: Models and Dynamic Embeddings.* In Proc. of Principles of Programming Languages, 1999.
5. R. Sion, M. Atallah and S. Prabhaker. *Resilient Information Hiding for Abstract Semi-Structures.* In Proc. of the IWDW, 2004
6. S. Inoue et al. *A Proposal on Information Hiding Methods using XML.* In the First NLP and XML Workshop.
7. M. Atallah, R. Sion and S. Prabhakar. *Watermarking non-media content.* In the the CERIAS Security Symposium, 2001.
8. UN/CEFACT and OASIS. *ebXML - Electronic Business using eXtensible Markup Language.* In http://www.ebxml.org/.
9. Y. Li, V. Swarup and S. Jajodia. *Constructing a Virtual Primary Key for Fingerprinting Relational Data.* In Proc. of the 2003 ACM workshop on Digital rights management, 2003.
10. C Fellbaum. *WordNet An Electronic Lexical Database.* The MIT Press, 1998.