J Real-Time Image Proc (2018) 14:183–192
https://doi.org/10.1007/s11554-016-0647-2

SPECIAL ISSUE PAPER

Two Zero-Watermark methods for XML documents

Quan Wen[1] · Yufei Wang[2] · Peng Li[3]

Received: 15 April 2016 / Accepted: 8 October 2016 / Published online: 24 October 2016

Abstract As XML files are less redundant and readily reorganized, it is really difficult to design a XML watermarking scheme which can get a trade-off between robust and invisible. However, this trade-off can be achieved by the Zero-Watermark method. In this paper, two Zero-Watermark methods are designed and tested for XML documents. One is XSLT-related method which is designed with embedding extra codes in XSLT file to serve as sort of copyright function. Another uses the functional dependency of XML file as feature for Zero-Watermark. Experiment results show that both methods have good real-time performances. Experiment results show that Zero-Watermark algorithm with functional dependency can resist selection attacks, alteration attacks, reorganization attacks and compression attacks.

Keywords Digital watermarking · Extensible Markup Language (XML) · Functional dependency · Zero-Watermark

✉ Quan Wen
wenquan@jlu.edu.cn

Yufei Wang
yufei-522@hotmail.com

Peng Li
lphit@163.com

[1] Jilin University, Changchun, China

[2] Jilin Normal University, Siping, China

[3] North China Electric Power University, Baoding, China

1 Introduction

XML (Extensible Markup Language) is now as important for the Web as HTML. It becomes the foundation of the Web due to its two important natures. One is using plain text to encode a hierarchical set of information. The other is using verbose tags to allow the document understanding without using any special reader or interpreter. Now XML is widely applied in information exchange, web service, Business-to-Consumer (B2C), Business-to-Business (B2B) and various areas such as recent telemedicine applications [1]. Anyway, XML is everywhere.

Meanwhile, the security of XML documents has become one of the most important issues in Web applications with XML. In fact, XML is not a new born thing, and XML-based security standards already exist and are relatively mature. For example, XML signature provides the integration and the non-repudiation; XML encryption provides the confidentiality for XML messages; SAML (Secure Assertion Markup Language) assures the identification. However, these traditional cryptographic mechanisms are priori, for which a posteriori technique is required [2]. For example, once the encrypted data are decrypted, the data of XML become clear and are no longer under the encryption protection. Therefore, as a posteriori protection mechanism, digital watermarking has gained increasing popularity as a supplement to traditional encryption security mechanisms of XML.

Since 1990s, a considerable amount of research efforts has been invested in the techniques of watermarking multimedia data such as images, audios and videos [3]. From the point view of practice, a trade-off has to be made between the robustness and the invisibility for watermarking schemes. Briefly speaking, the more distortions introduced in the watermarking process, the more likely a watermark can be detected and attacked. To get this trade-off, the concept of Zero-Watermark was proposed [4]. Zero-Watermark is an approach which does not require embedding the watermark information into the original data. Instead, a watermark in Zero-Watermark is constructed or generated based on the content of the original data.

In contrast, relational database and XML are less redundant and readily reorganized. These features are the reason why it is much more difficult to design a robust and invisible watermarking scheme for XML documents than for multimedia data. If the watermark embedding process will introduce distortion to the cover database, these watermarking schemes are called *distortion-based* [5–7]. Otherwise, those watermarking schemes for which any distortion will not be introduced in the cover database are classified into *distortion-free* scheme*s* [8–10]. The purpose of designing *distortion-free* scheme*s* is to address the trade-off between robust and invisible. However, to the best of our knowledge, most distortion-free methods are based on reordering, which are intrinsically fragile. The lack of primary key makes the reordering method very difficult to deploy. That is the main reason why there is almost no distortion-free method for XML documents.

The key point of Zero-Watermark scheme is to extract the unchangeable feature from the cover data, while the distortion-free method only considers how to encode watermark information without distortion. Thus, it is possible to design XML Zero-Watermark scheme which can satisfy robustness and invisibility at the same time. In this paper, we firstly analyze the query-preserving ability in Zero-Watermark design, which most of distortion-free methods do not concern. After that, two Zero-Watermark methods are proposed and verified with experiments.

The rest of this paper is organized as follows. In Sect. 2, we introduce the related work; we focus on some theoretical analysis for Zero-Watermark XML schemes in Sect. 3; two detailed Zero-Watermark methods for XML documents are described in Sect. 4; we show the experiment results in Sect. 5 and the final section is a conclusion and future work.

2 Related works

Digital watermarking problem is not a new topic and has been studied since 1990s. However, considering the redundancy and data reorganization problems in relational databases and XML documents, it was not until 2003 that the first watermarking research work for relational database was proposed by Agrawal et al. [11] (also known as AHK algorithm). At the same time, S. Inoue et al. [12] proposed the first special information hiding suggestion for XML documents. But, S. Inoue only proposed the scheme from the view of information hiding, which was not designed for the watermarking in copyright protection. Thus, his research did not get too much attention. AHK scheme embeds watermarks within LSB of numeric attributes. This scheme assumed that merchants and buyers can tolerate a small amount of errors in those attributes, while the introduced errors substantially reduced the value of the data. Based on AHK scheme, Ng and Lau extended it to watermark XML data and deployed it on XML compression system [7]. However, Ng and Lau's scheme was the 1-bit embedding watermarking, which did not fully take advantage of XML documents usability.

After that, many works were focused on the distortion durability problem, since most of relational databases cannot even tolerate tiny distortion caused by watermark embedding. Some researchers tried to minimize the distortion [6, 13, 14]. Others even proposed distortion-free methodologies [8–10]. To the best of our knowledge, there is no specific distortion-free scheme designed for XML documents. One of the main reasons is that most researchers think their watermarking schemes for relational database could be naturally extended to XML documents. However, XML documents are different from relational database. For instance, XML is self-describing and hierarchical. Thus, XML watermarking needs a special design concern. In addition, many database watermarking methods are designed by simply using the information of pure numerical data tuples in databases. However, many data in XML documents are categorical. The categorical data are more difficult to be watermarked than the numerical data. Furthermore, it is difficult to embed watermark into XML documents even with tiny distortion, while watermarked XML documents keep the query result preservation [15]. To sum up, a robust and distortion-free watermarking mechanism is very difficult to realize, but it is essential to XML documents. Therefore, Zero-Watermark scheme as a promising solution is worth to be studied in detail.

3 Introduction of Zero-Watermark

In order to better describe the concept of Zero-Watermark conception, architecture of proposed methods is given first, and then, we discuss some theoretical issues.

3.1 Zero-Watermark mechanism architecture

Generally speaking, the way that Zero-Watermark achieves copyright can be described in Fig. 1. Different from most of watermarking schemes, no embedding phase can be found. Instead, the key of Zero-Watermark is the generation algorithm G which can generate the unique and

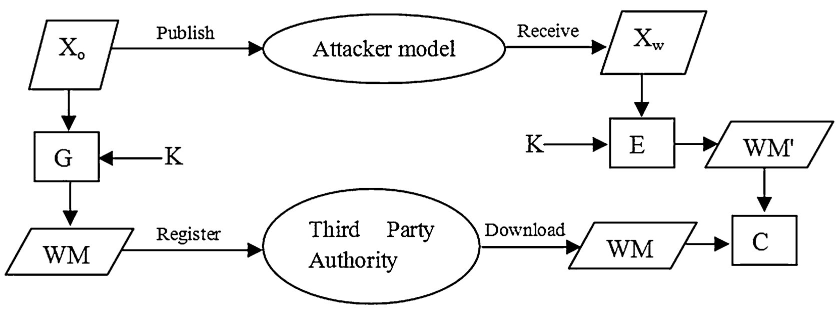


Fig. 1 General architecture of Zero-Watermark mechanism

unchangeable feature from the original media X_O to make the watermark WM. The watermark WM can be extracted or generated again after an attack, if the feature of X_O generated by G cannot be easily removed without damaging the business value of X_O. In fact, the watermark generation algorithm G is the same as the watermark extraction algorithm E. Both G and E are designed to get the unchangeable features of XML documents. Also, it is shown in Fig. 1 that Zero-Watermark naturally is a blind watermarking scheme.

3.2 Zero-Watermark theoretical analyze

From the theoretical point of view, a watermarking problem usually can be described by the following formula:

$$X_W = X_O \otimes \alpha \bullet \mathrm{WM} \tag{1}$$

X_o is the original data; X_W is watermarked data; $\otimes$ indicates any kind of the embedding algorithm; α is the embedding strength, which is the key of getting trade-off between robustness and invisibility. Technically, Zero-Watermark is different from any traditional watermarking scheme. It also can be explained by Eq. (1) under the condition let parameter α be 0. Therefore, Zero-Watermark namely is the 0-embedding strength watermarking system.

From the query-preserving point of view, Zero-Watermark can be theoretically analyzed by the theory provided by Gross-Amblard [15]. Gross-Amblard did some quite theoretical work based on VC-dimension in the field of computational learning theory for analyzing the problem of query-preserving in relational databases and XML documents. Taking the formalism of watermarking problem based on VC-dimension, a watermarking problem can be represented by a (l, d)-watermarking protocol in which l is the embedding bit and d is the distortion caused by watermark embedding. In theory, Zero-Watermark protocol can be taken as a kind of (l, 0)-watermarking protocol.

Gross-Amblard proposed an important remark in his work which was 'unbounded VC-dimension is not sufficient' [15]. To prove this remark, he used a ($|W|/4$, 0)-watermarking protocol that is a kind of Zero-Watermark. Using this distortion-free watermark protocol, he proved that the unbounded VC-dimension is not sufficient as this kind of watermarking protocol has shattered (VC(ψ, ς) = $|W|/2$).

In theory, the Zero-Watermark idea can cause a certain theoretical vulnerability to the proof of Theorem 3.5 in Gross-Amblard's paper [15]. As the Zero-Watermark protocol does not embed at all, the distortion function is represented as follows:

$$h(|W|, \varepsilon) = \frac{|W|^{1-q\varepsilon}}{2\ln|W|}. \tag{2}$$

Because the $|W|$ embedding length of bit is also zero under the Zero-Watermark circumstance, function h cannot be the increasing function. Then, there is no contradiction under the condition of $if \quad \forall\varsigma \in \kappa, \quad \mathrm{VC}(\psi, \varsigma) = |w^{\varsigma,\psi}|$ which is the key to prove the Theorem 3.5. In other words, for a XML tree structure, even if its VC-dimension is unbounded, Zero-Watermark scheme still could be a possible scalable watermarking protocol. Nevertheless, it does not mean the theory proved by Gross-Amblard is not correct. Indeed, it simply cannot apply directly to Zero-Watermark which is a very particular case without any embedding procedure.

4 Zero-Watermark scheme

In this section, two methods implementing Zero-Watermark idea are discussed in detail. One is designed with XSLT in which the watermarking functional code is embedded. The other is achieved by using *functional dependency* to get the unique and usable features of XML documents.

4.1 Zero-Watermark by XSLT

Basically, XML is also a kind of relational database. However, XML is not only a database, but has some special designs for its functional purpose. Nonetheless, these designs in XML can lead us to find some possible opportunities to design Zero-Watermark scheme associated with the XML web service system. Then, we found XSLT which stands for Extensible Stylesheet Language Transformation.

XSLT transforms an XML document into a different structure. XSLT is used for two common purposes: (1) To convert XML format into a presentation-specific one, such as HTML; (2) To convert the format understood by one application into the structure required by another. According to these two situations, XSLT is a very important part of XML package and is needed in almost every XML files in Internet. If you want your browser to show the data in XML files, extra files such as XSLT are always needed. Therefore, we can embed watermark code in the XSLT files meanwhile not changing the associated XML files. Technically, this XSLT method can satisfy the requirement of Zero-Watermark conception.

According to the idea of the Zero-Watermark, we can extract the features of XML documents by using extra codes in XSLT. That means an XSLT document with copyright mechanism is different from normal XSLT documents. We encode some extra codes for watermarking purpose in the special XSLT file. For example, '<xsl: template>' can be used for the watermarking purpose. '<xsl: template>' is the bedrock of XSLT and has two main features. It details what items from the source document should be handled and uses its content to specify what should be added to the output when it is executed. Usually, there can be many kinds of templates in the XSLT files, so we can hide the watermark code in some of the templates codes. Furthermore, a switch for watermarking can be set the XSLT. The mark switch is set to off for the normal case. When the state of the mark switch is on, a copyright mark can be found in the HTML through browser. The Zero-Watermark with XSLT explained above is shown as a framework in Fig. 2.

Taking the people.xml in Fig. 3 for example, the semantic feature in it could be the names of these celebrities. In the XSLT file accompanying with people XML document, we could embed the watermarking codes which extracts names of people as features in XML documents. The watermarking code fragments are given in Fig. 4.

It is shown in Fig. 4, the XSLT code element of <xsl:choose> is used to decide when to show the watermark copyright sign. The variable 'IsMarked' plays the

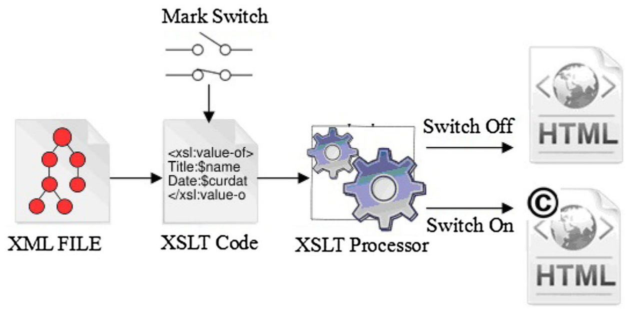


Fig. 2 Zero-Watermark framework via XSLT

role of watermark switch in Fig. 2. The variable 'IsMarked' add by 1 when some names are detected the same as the original XML files. In many cases, just a few lines of the watermarking code can get work done. Because a XSLT file in application normally contain thousands of code lines, the extra codes for watermarking are difficult to be noticed.

4.2 Zero-Watermark with functional dependency

Zero-Watermark with XSLT can meet the requirement of Zero-Watermark without modifying original XML file. However, the problem of this method is that different XSLT codes have to be designed for different XML file types. Considering the versatility problem, we design Zero-Watermark method by *functional dependency*.

Like traditional database, XML also has a redundancy problem in which *functional dependency* plays an important role. Functional dependency of XML tree is defined as follows:

Given an XML tree T that conforms to a schema S, a functional dependency (FD) is an expression of the form $P_1 \rightarrow P_2$ where $P_1, P_2 \subset paths(S)$, T satisfies $P_1 \rightarrow P_2$. If for any pair of tuples d_1, d_2 in the flat table FT(T), $d_1 \cdot P_1 = d_2 \cdot P_1$ implies $d_1 \cdot P_2 = d_2 \cdot P_2$ [16].

For example, the FDs in book.xml are listed as follows:

$$book/title \rightarrow book;\ book \rightarrow book/title;\ book \\ \rightarrow book/rating;\ book/title \\ \rightarrow book/publisher,\ book/editor.$$

Sometimes, an XML FD can also be represented as a triple $\langle C_p, LHS, RHS \rangle$, in which, C_p denotes a tuple tree; LHS (left-hand side) is referred to P_1; RHS <right-hand side> is referred to P_2 [17].

Generally speaking, FDs of XML are used to normalize the XML documents to eliminate the duplicates of XML. As FDs work at the semantic level, it can serve as an invariant and unique feature which is suitable to be used as the Zero-Watermarking bits. Because most of attacks cannot influence the semantic information of XML, a Zero-Watermark method based on FDs can be designed with high robustness. The details of algorithm are described in the generation and detection processes, respectively.

4.2.1 Zero-Watermark generation algorithm

The construction algorithm of the Zero-Watermark with FDs is described in Fig. 5. *Discover*XFD is an efficient algorithm to find XML FDs and data redundancies proposed by C. Yu et al. [17]. It is used here for generating the Zero-Watermarking bits information.

In Fig. 5, the *Discover*XFD is an improved version algorithm to find the FDs in XML structures [17]. How to

```
<people>
 <person bomDate="1874-11-30" diedDate="1965-01-24">
  <name>Winston Churchill</name>
  <nationality>British</nationality>
  <description>
    Winston Churchill was a mid 20th century British politician who
   became famous as Prime Minister during the Second World War.
  </description>
 </person>
<person bomDate="1917-11-19" diedDate="1984-10-31">
  <name>Indira Gandhi</name>
  <nationality>India</nationality>
    <description>
    Indira Gandhi was India's first female prime minister and was assassinated in 1984.
    </description>
  </person>
  ...
  </people>
```

Fig. 3 people.xml

```
<xsl:choose>
  <xsl:when test=" $IsMarked > 1 ">
       show XML with watermarking sign in the browser
  </xsl:when>
  <xsl:otherwise>
       show the normal XML in the browser
  </xsl:otherwise>
```

Fig. 4 Watermarking codes in the XSLT file

effectively find FDs in XML is still an open to further research, while Zero-Watermark with FDs can improve its performance with the development of *Discover*XFD algorithms. At the line 2, before call *Discover*XFD function, XML data set is converted to hierarchical representation which is suitable input for *Discover*XFD function. At the line 4, FDs, the output of *Discover*XFD have the strings format. At the line 5, the function Get_Binary is called to convert the FDs to binary format that can be the Zero-Watermark bits WM_0. The detail of Get_Binary is given from the line 7 to line 10. At the line 10, we encrypt Zero-Watermark bit with the secret key *K*. It is concerned for security issue.

Considering Zero-Watermark is robust watermarking scheme, hash function is not used here because a tiny change can cause the major change of the output of hash function.

4.2.2 Zero-Watermark detection algorithm

Different from the traditional watermark scheme, detection process of Zero-Watermark is not the reverse process of watermark insertion. Its detection process

```
Algorithm: GenZW
Input:    XML Data Set XD, Secret Key K
Output:  zero watermark bit string WM_0
  1)   proc GenZW(XD)
  2)      Convert XD to Hierarchical Representation R_p
  3)      Call DiscoverXFD
  4)          FDs = DiscoverXFD(R_p)
  5)          WM_0=Get_Binary(FDs, K)
  6)   end proc

        Function Get_Binary(FDs, K)
  7)     for each FDs
  8)          String S_FD = letters extract ( FDs.LHS, FDs.RHS)
  9)       N_b = String2Binary(S_FD)
 10)       WM_0 = Encode(N_b, K)
```

Fig. 5 Zero-Watermark generating algorithm base on FDs

functions the same as the process of Zero-Watermark generation. But, the input XML document is different from the one at generation phase. At the detection scenario, the inputs XML documents can be modified or attacked, but the detection is still successful if the feature used to construct Zero-Watermarking bits is not influenced under attack.

As shown in Fig. 6, the detection process also needs to generate the Zero-Watermark bits just like the generating process. However, the input for detection process is XD_w, which is suspected to violate the copyright and is probably modified or attacked. At the line 2, XD_w also need to be converted to hierarchical representation for *Discover*XFD function just like the same thing in GenZW. At the lines 4 and 5, *Discover*XFD and Get_Binary are called again to extract the $\mathrm{WM_m}$ from XD_w. At the line 6, we use the Compare function to check if there is Zero-Watermark information in XD_w. The detail of Compare function is described from line 8 to line 13. The key of Compare function is that the similarity between $\mathrm{WM_m}$ and $\mathrm{WM_o}$ is computed at the line 8. The value of 'T' used in the function *Compare* is the detection threshold. T equals to 1 if $\mathrm{WM_w}$ is totally equivalent to $\mathrm{WM_o}$.

At the end of this section, a simple scenario is presented to illustrate the application of Zero-Watermark with FDs. The FD *book/editor* → *book/publisher* from book.xml in Fig. 7 satisfies the definition mentioned earlier, because we assume that each editor only works for one publisher. Hence, we get the keyword pair which is (editor, publisher). With this pair, we get watermark bits which are supposed to be '1001100011' by algorithm GenZW in Fig. 5. Meanwhile, the Zero-Watermark information will be registered or stored in a trust agency like described in Fig. 1. The purpose of store in a third authority party is that the original watermark can be downloaded for comparing

```
<book publisher="sams">
<title>securing web services with WS-Security</title>
<author>Jothy Rosenberg</author>
<author>David Remy</author>
<editor>Todd Green</editor>
<rating>40</rating>
</book>
<book publisher="mcgrawhill">
<title>XML security</title>
<author>Blake Dournaee</author>
<editor>Betsy Manini</editor>
<rating>47</rating>
</book>
…
</book>
```

Fig. 7 book.xml

with the extracted watermark during the detection phase. When it goes to the watermarking detection phase, the book.xml can be modified by various malicious attacks. However, the FD: *book/editor* → *book/publisher* can survive under attacks, because the assumption that 'each editor only works for one publisher' still holds. So, the Zero-Watermark bits '1001100011' can be constructed successfully again and equal to the original one, which means the Zero-Watermark with FDs works at this scenario.

5 Experiments

We have implemented two experiments to verify the above-mentioned Zero-Watermark methods. All of the simulation procedures are executed on an Intel Core i3 CPU running at 2GHz with 2 GB of physical memory.

Fig. 6 Zero-Watermark by FDs detection algorithm

```
Algorithm: DetZW
Input:    XML Data Set to be detected $XD_w$, Secret Key $K$, Original Watermark $WM_o$
Output:   Yes or No answer if $XD_w$ have copyright
  1)   proc DetZW($XD_w$)
  2)        Convert XD to Hierarchical Representation $R_{pw}$
  3)        Call DiscoverXFD
  4)             FDs = DiscoverXFD($R_{pw}$)
  5)             $WM_w$=Get_Binary(FDs, $K$)
  6)        Compare($WM_w$, $WM_o$)
  7)   end proc
        function Compare($WM_w$, $WM_o$)      /* 1/2<T< 1 */
  8)        Sim = $WM_W \bullet WM_O / ||WM_W||$
  9)        if Sim > T, then
 10)           Ans = Yes;
 11)        else
 12)           Ans = No;
 13)        print Ans
```

5.1 Experiment of Zero-Watermark by XSLT

An XML professional editor '*Oxygen XML Editor*' is used to show an example of Zero-Watermark by XSLT. The XSLT file with watermark extra code in it is edited by Oxygen software. The XML file of people in Fig. 3 is used to simulate the Zero-Watermark by XSLT. The following code fragment is the head of the XSLT file.

As shown in Fig. 8, the variable '*markswitch*' is used to decide whether or not the copyright information is visible in browser. The names in people.xml file are used as a feature of Zero-Watermark and are collected in an outside file as names.xml. Because there are many files included in a real XSLT file, it is difficult to know which head file including data is used for watermarking. The include file for Zero-Watermark information, such as names.xml, hide in many include files, which will lead a malicious attacker to cost more to find the flaw of Zero-Watermark.

Set the state of '*markswitch*' as '*off*' or '*on*', we get the normal html and copyright html showed in Fig. 9. From Fig. 9, we can notice two different places. One is at the head title; the other is at the dates of birth and death. Compared with the copyright sign '@Quan', the date format difference is more difficult to notice. In other words, '@Quan' is visible watermarking; and the difference date format can be sort of semi-invisible watermarking which has better security. The experiment is a simple example to verify the availability of Zero-Watermark with XSLT. It is easy to find more practical methods to show copyright sign for a real complicated XSTL file.

5.2 Experiment of Zero-Watermark by FDs

To test the Zero-Watermark method by FDs, we use the structure XML document which is the same as the one in Fig. 7, the book.xml. We encode more than 620,000 publication books with the data size 55.6 Meta bytes.

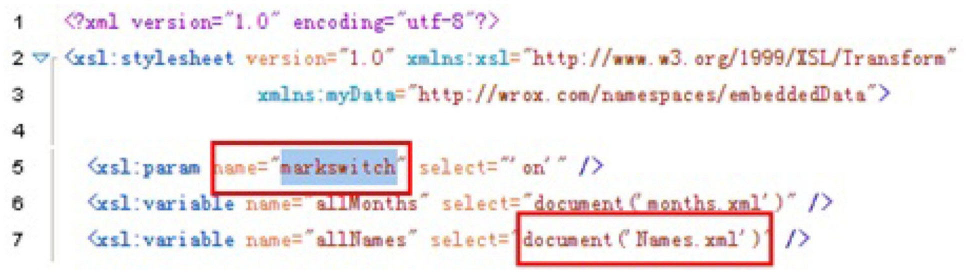
Here, we focus on the resistance of Zero-Watermark. Four kinds of attacks are executed to test robustness, which are selection attack, alteration attack, reorganization attack and compression attack. The fourth attack compression attack is very special to XML watermarking, which most of XML watermarking mechanisms do not test.

For a selection attack, an attacker selects a part of data that satisfy the intended purpose and discard the rest. We simulate the attack through random selection rate from 10 to 80 %. It is shown in Fig. 10a that all of the similarity values between extracted Zero-Watermark and original Zero-Watermark are above the 0.8. It means the detection is successful when the similarity value is up to 0.6. Moreover, there are distortions at the two points of 50 and 70 %. As the selection is random, the part of selected data exactly contributes a lot to the FDs at the two points, which exerts an influence on the detection result.

The second attack test aims to study the resilience of Zero-Watermark by FDs against alteration attack. To simulate the attack, we randomly select some nodes, and modified their *title*, *author* and *rating* attributes in order to ruin the Zero-Watermark detection. The results in Fig. 10b are got under the alteration attack with proportions from 10 to 80 %. As the Zero-Watermark is generated by FDs features which do not relate to the nodes value, the resilience test results show that Zero-Watermark with FDs has a good performance under an alteration attack.

The third set of experiments is designed to study the resilience to the reorganization attack which is the challenge to most of existing watermarking methods. We use three different reorganization models to simulate the resilience. Three models, respectively, change <*publisher*>, <*title*>, <*rating*> to be the father node. All of the three models use the change proportions from 10 to 90 %. Figure 10c shows that the <*rating*> organization model can most effectively attack the Zero-Watermark system, as the <*rating*> elements relate to most part of FDs in the book.xml XML file.

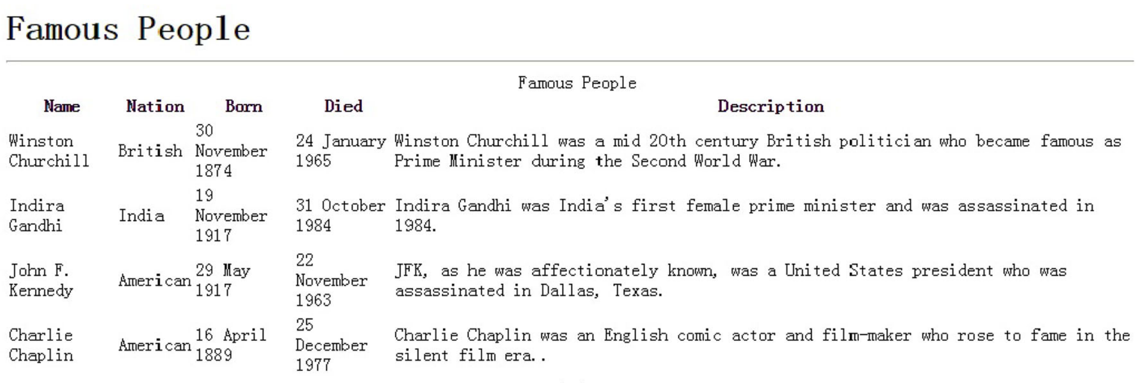
The last set of experiments is verified for studying the resilience facing compression attack. Compression attack is very common in multimedia watermarking mechanism, but most of XML watermarking mechanisms do not consider this attack type. However, the compression attack is common since most of XML file need compression to meet the redundancy problem. Three different compressors *XQzip*, *XMill* and *XGind* are used as compressor to carry out the XML compression [7]. Different proportions of XML file are selected and compressed via *XQzip* to test the resilience for compression. The results are shown in Fig. 10d. As FDs work at the semantic level, the compression on data cannot



```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:myData="http://wrox.com/namespaces/embeddedData">

  <xsl:param name="markswitch" select="'on'" />
  <xsl:variable name="allMonths" select="document('months.xml')" />
  <xsl:variable name="allNames" select="document('Names.xml')" />
```

Fig. 8 Head code in the XSLT file

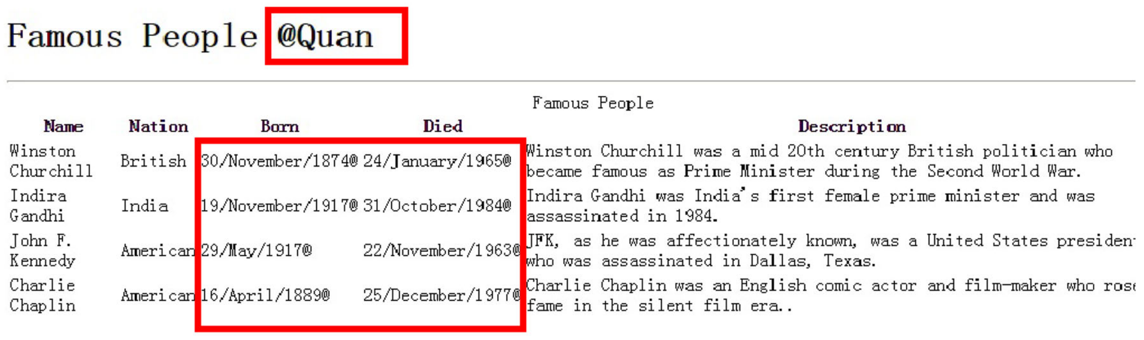


Famous People

Famous People

Name	Nation	Born	Died	Description
Winston Churchill	British	30 November 1874	24 January 1965	Winston Churchill was a mid 20th century British politician who became famous as Prime Minister during the Second World War.
Indira Gandhi	India	19 November 1917	31 October 1984	Indira Gandhi was India's first female prime minister and was assassinated in 1984.
John F. Kennedy	American	29 May 1917	22 November 1963	JFK, as he was affectionately known, was a United States president who was assassinated in Dallas, Texas.
Charlie Chaplin	American	16 April 1889	25 December 1977	Charlie Chaplin was an English comic actor and film-maker who rose to fame in the silent film era..

(a)



Famous People @Quan

Famous People

Name	Nation	Born	Died	Description
Winston Churchill	British	30/November/1874@	24/January/1965@	Winston Churchill was a mid 20th century British politician who became famous as Prime Minister during the Second World War.
Indira Gandhi	India	19/November/1917@	31/October/1984@	Indira Gandhi was India's first female prime minister and was assassinated in 1984.
John F. Kennedy	American	29/May/1917@	22/November/1963@	JFK, as he was affectionately known, was a United States presiden who was assassinated in Dallas, Texas.
Charlie Chaplin	American	16/April/1889@	25/December/1977@	Charlie Chaplin was an English comic actor and film-maker who ros fame in the silent film era..

(b)

Fig. 9 Browser show with XSLT watermarking code, **a** with watermark switch off, **b** with watermark switch on

influence the Zero-Watermark efficiency. However, *XQzip* has a better attack performance as it has the higher compression ratio.

Then, we compare the robustness performance between Zero-Watermark by FDs and previously proposed relational database watermarking methods with distortion minimization [6, 13, 14]. Four kinds of attacks in Fig. 10 are used under very strict conditions in order to test the robust limit in those methods. For compression attack, the XQZip compressor is used. If the similarity of detection is over 0.6, then the answer is 'Yes'; otherwise it is 'No'. As the previous methods used for comparison are related to relational database, the book.xml in Fig. 7 has to be transformed into relational data format before it is used for test. The comparison results are given in Table 1. It turns out that the robustness advantage of Zero-Watermark with FDs method is obvious.

5.3 Real-time experiment

As Zero-Watermark saves the embedding procedure, it has a good real-time performance. The time costs of two methods are tested by replicating data in book and people XML files. The experiment results are shown in Table 2.

For Zero-Watermark with XSLT, a time–cost comparison between the normal display and the copyright display is listed. For Zero-Watermark with FDs, a time–cost comparison between generation procedure time and detection procedure time is listed. As shown in Table 2, Zero-Watermark with XSLT costs a little more execution time when the file size arrives at 22.4 and 44.8 Mbytes. Zero-Watermark with FDs takes a little longer than Zero-Watermark with XSLT, but there are not many differences between generation procedure and detection procedure.

6 Conclusion

In this paper, we verified that Zero-Watermark is a good idea to meet the challenge of XML watermarking. Two real-time and robust methods from Zero-Watermark conception are designed and tested. Zero-Watermark with XSLT adds the watermarking codes into XSLT file. Zero-Watermark with FDs uses the functional dependency as a feature to construct Zero-Watermark information. Both algorithms match the conception of Zero-Watermark, which does not modify the original XML file. Various attacks including compression attack are studied for the

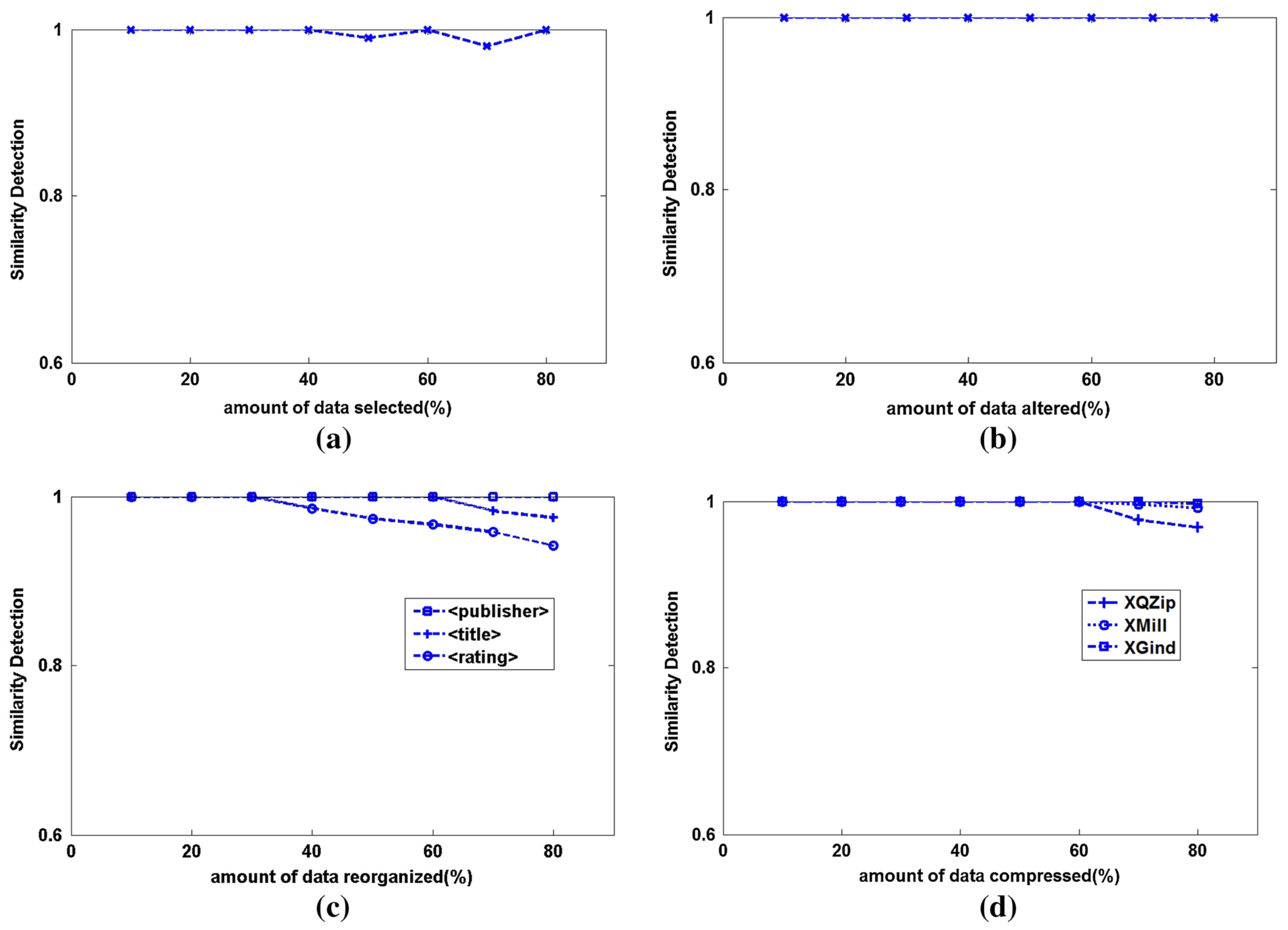


Fig. 10 Resilience to various attacks: **a** selection attack, **b** alteration attack, **c** reorganization attack, **d** compression attack

Table 1 Robustness comparison between Zero-Watermark with FDs and previous methods

Attack type	Watermark detection result			
	Ref. [13]	Ref. [14]	Ref. [6]	Zero-Watermark by FDs
Selection with 80 %	No	Yes	Yes	Yes
Alteration with 90 %	Yes	Yes	Yes	Yes
Reorganization with 90 %	No	No	Yes	Yes
Compression with 80 %	No	No	No	Yes

Table 2 Execution time of two Zero-Watermark methods with different file size

File size (MB)	Time (s)			
	XSLT method		FDs method	
	Normal	Marked	Generation	Detection
5.6	0.066	0.066	0.1275	0.1273
11.2	0.067	0.067	0.2404	0.2401
22.4	0.1240	0.1241	1.0123	1.0102
44.8	1.0596	1.2673	2.6342	2.5947

Zero-Watermark by FDs. Although Zero-Watermark with XSLT is not a general algorithm like Zero-Watermark with FDs, it is still a promising practical method for some kind of XML documents for which a special copyright protection scheme is need.

For future work, more available features in XML file for Zero-Watermark scheme are needed to be found. Also, more practical programming methods are required for Zero-Watermark with XSLT.

Acknowledgments Funding was provided by Youth Science Foundation of JiLin Province (Grant No. 20100183).

References

1. Wu, J.: A framework for learning comprehensible theories in XML document classification. IEEE Trans. Knowl. Data Eng. **24**(1), 1–14 (2012)
2. Franco-Contreras, J., Coatrieux, G.: Robust watermarking of relational databases with ontology-guided distortion control. IEEE Trans. Inf. Forensics Secur. **10**(9), 1939–1952 (2015)
3. Petiteolas, F.A.R., Anderson, R.J., Kuhn, M.G., Information hiding—a survey. In: Proceedings of IEEE, pp. 1062–1078 (1999)
4. Wen, Q., Sun, T.-F., Wang, S.-X.: Concept and application of zero-watermark. Tien Tzu Hsueh Pao/Acta Electron. Sin. **31**(2), 214–216 (2003)
5. Sion, R., Atallah, M., Prabhakar, S.: Rights protection for relational data. IEEE Trans. Knowl. Data Eng. **16**(6), 1–17 (2004)
6. Kamran, M., Suhail, S., Farooq, M.: A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints. IEEE Trans. Knowl. Data Eng. **25**(12), 2694–2707 (2013)
7. Ng, W., Lau, H. L.: Effective approaches for watermarking XML data. In: DASFAA, International symposium on database systems for advanced applications, pp. 68–80 (2005)
8. Li, Y., Deng, R.H.: Publicly verifiable ownership protection for relational databases. In: ASIACCS '06 proceedings of the 2006 ACM symposium on information, computer and communications security, pp. 78–89 (2006)
9. Bhattacharya, S., Cortesi, A.: A distortion free watermark framework for relational databases. In:Proceedings 4th international conference on software and data technology, ICSOFT 2009, Sofia, Bulgaria, pp. 229–234 (2009)
10. Camara, L., Li, J., Li, R., Xie, W.: Distortion-free watermarking approach for relational database integrity checking. Mathematical problems in engineering, pp. 1–10. Hindawi Publishing Corporation, Cairo (2014)
11. Agrawal, R., Haas, P.J., Kiernan, J.: Watermarking relational data: framework, algorithms and analysis. VLDB J. **12**(2), 157–169 (2003)
12. Inoue, S., et al.: A proposal on steganography method using XML. In: Proceedings 2002 Symposium cryptography and information security (SCIS2002), pp. 301–306 (2002)
13. Shehab, M., Bertino, E., Ghafoor, A.: Watermarking relational databases using optimization-based techniques. Trans. Knowl. Data Eng. **20**(1), 116–129 (2008)
14. Lafaye, J., Gross-Amblard, D., Constantin, C., Guerrouani, M.: Watermill: an optimized fingerprinting system for databases under constraints. Trans. Knowl. Data Eng. **20**(4), 532–546 (2008)
15. Gross-Amblad, D., Saclay, I.: Query-preserving watermarking of relational database and XML documents. ACM Trans. Database Syst. **36**(1), 1–24 (2011)
16. Zhou, X., Pang, H., Tan, K.L.: Query-based watermarking for XML data. In: Proceedings of the 2nd ACM symposium on information, computer and communications security, pp. 253–264 (2007)
17. Yu, C., Jagadish, H.V.: Efficient discovery of XML data redundancies.In: VLDB '06 Proceedings of the 32nd international conference on very large data bases, pp.103–114 (2006)

Quan Wen received the M.S. degree in Communication and Information System from Jilin University in 2003, and the Ph.D. degree in 2005. He is currently a Lecturer in College of Communication Engineering at Jilin University. His research interests include digital watermarking and tactile display in HCI.

Yufei Wang received the M.S. degree in Communication and Information System from Jilin University in 2006, and the Ph.D. degree in 2010. She is currently a Lecturer in Department of Communication Engineering at Jilin normal University. Her research interests include information security and electromagnetic environment analysis.

Peng Li received the B.S. degree in Applied Mathematics from North China Electric Power University in 2004, and the M.S. degree in Computational Mathematics from Harbin Institute of Technology in 2006. He received his Ph.D. degree in Computer Application Technology from Harbin Institute of Technology, China, in 2012. He is currently a Lecturer in Department of Mathematics and Physics at North China Electric Power University. His research interests include secret image sharing, visual cryptography and information hiding.