Enhancing XML Zero-Watermarking Robustness Using Usability Queries and Functional Dependencies

Benedek Székács[1]

Supervisors: Supervisors: Zeki Erkin[1]**, Devris Isler**[2]

[1]**EEMCS, Delft University of Technology, The Netherlands**
[2]**IMDEA Networks Institute, Universidad Carlos III de Madrid, Spain**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Benedek Székács *Email: b.szekacs-2@student.tudelft.nl*
Final project course: CSE3000 Research Project
Thesis committee: Zeki Erkin, Devris Isler, Asterios Katsifodimos

Abstract

In the digital era, XML data is fundamental for various applications, requiring robust methods to ensure data integrity and security. Traditional digital watermarking techniques face challenges due to XML's hierarchical structure. Zero-watermarking, which derives a watermark from inherent data properties without altering the data, offers a promising alternative. This paper enhances zero-watermarking robustness for XML by integrating usability queries to filter functional dependencies (FDs) based on their relevance. This approach maintains watermark integrity even when some data elements are modified, provided the data's usability is preserved. The proposed method increases resilience against targeted attacks altering semantic information while preserving usability. Key contributions include incorporating usability queries into the FD-based zero-watermarking process, generating cover ranges for usability queries, and empirically testing the method's robustness.

1 Introduction

In the digital era, XML (Extensible Markup Language) has become a cornerstone for data representation and exchange across diverse online platforms. Its structured format supports complex data interactions and services in areas ranging from online banking to healthcare systems. Particularly in recent years, the adoption of XML has expanded into large-scale applications, managing vast amounts of data that drive major business and governmental operations globally. As XML data often contains sensitive and valuable information, ensuring its security and integrity is a paramount concern.

Digital Watermarking, in the traditional sense, is the practice of embedding secret information into media content, such that it is unperceivable and does not affect the usability of the data. The main purpose of a watermark is to prove ownership of the data and prevent its illegal distribution or tampering. From the 1990s, techniques for watermarking multimedia data (image, audio, video) have been widely explored. They exploit the fact that such data has a high noise tolerance making it easier to hide extra information in it. More recent studies extended watermarking to non-media type data domains such as software, machine learning models or relational databases. While these methods can be adapted to embed and detect the watermark bits in the isolated XML data elements, they are not resilient to structural reorganization attacks.

Zero-watermarking, which derives a watermark from the inherent properties of data without altering it, offers a promising solution. Recent advancements propose creating identifiers for XML nodes using usability queries, allowing for resilient watermarking against structural changes. This paper builds on these ideas by integrating usability queries into the functional dependency-based zero-watermarking process, enhancing robustness against targeted attacks that alter semantic information while preserving usability.

The primary focus of this research is to enhance the robustness of zero-watermarking techniques for XML data by incorporating user-defined usability queries. By leveraging functional dependencies (FDs) extracted from XML data and filtering them based on their relevance to these usability queries, we ensure that the watermark remains intact even if some data elements are modified, provided the data's usability is preserved. This method not only improves the robustness of the watermark but also aligns with the growing need for secure and reliable data management in XML-based systems.
Our contributions include:

- Proposing novel context-based targeted attacks on XML watermarks.
- Proposing a novel integration of usability queries into the FD-based zero-watermarking process, enhancing the robustness of watermarks against targeted semantic attacks.
- Improving the resilience of FD-based Zero-Watermarking against Zero-out attacks.
- Implementing the DiscoverFD algorithm to extract functional dependencies from XML data.
- Conducting theoretical analysis and empirical testing to validate the effectiveness and robustness of the proposed method.

This work builds on the foundational algorithm for zero-watermarking proposed by Wen et al. which introduced two methods for watermarking XML documents without altering their original content: an XSLT-related method and a functional dependency-based method. Wen et al. assumed that attacks would not change the semantic information of the data, hence preserving functional dependencies. However, some functional dependencies might not be closely tied to the data's intended usability, making them vulnerable to attacks. By using the cover ranges of usability queries to filter the functional dependencies used in the watermark generation algorithm, we aim to increase resilience against attacks that target and delete unused functional dependencies.

The structure of this paper is as follows: Section 2 reviews related work on digital watermarking techniques for relational databases and XML, including the DiscoverXFD algorithm and Wen et al.'s zero-watermarking methods. Section 3 details the proposed method, including the integration of usability queries and the generation of zero-watermarks. Section 4 describes the implementation of the proposed method and the experimental setup. Section 5 presents the experimental results and analysis. Finally, Section 6 concludes the paper and discusses future research directions.

2 Related Work

There has been extensive research on digital watermarking since the 1990s. However the first work concerning relational data was only proposed in 2002 by Agrawal et al. [1] Their method (the AHK algorithm) embeds watermarks in numeric data by slightly modifying the least significant bits (LSB) of the values within acceptable error bounds.

This approach ensures that the watermark is resilient to typical database operations such as sorting, insertion, and deletion. Sion et al. [9] extended this idea by proposing watermarking techniques that consider various database constraints, including primary keys and foreign keys, ensuring that the watermarking process does not violate database integrity. Another significant contribution is the work by Li et al. [3], who presented a framework for watermarking categorical data. This approach involves embedding watermarks by modifying non-numeric data values while preserving the data's usability and statistical properties. There have been a number of works on relational databases since then and they all achieve various levels of resilience against delete, update and insertion attacks [5; 6]. These are also used against XML watermarks and we will include them in our experiments later in this paper. However these previous works do not address data reorganization attacks explained in section 3.

The transition from relational databases to semi-structured data formats like XML introduced new challenges and opportunities for digital watermarking. XML's hierarchical structure and widespread use in data exchange necessitate robust watermarking techniques that can withstand various forms of data manipulation. Expanding on the AHK method, Ng and Lau [4] introduced a robust watermarking technique for XML documents that embeds watermarks by modifying element tags and attributes. Their approach ensures that the watermark is embedded in a way that does not disrupt the document's usability or its ability to be correctly parsed by XML parsers. In [8], Sion et al. proposed an information hiding scheme for semistructured data, in which they identify each data element by the values of its adjacent elements, and hope the identifiers could survive through reorganization. Zhou et al.[13] proposed a query-based watermarking technique specifically designed for XML data. This method creates identifiers for XML nodes using usability queries, allowing the watermark to be resilient to structural reorganizations and redundancy removal attacks. The approach leverages the inherent redundancy in XML data, ensuring that the same identifier is generated for all copies of nodes, thereby maintaining the watermark's integrity even when the data structure is altered.

The work by Wen et al. [11] serves as a foundational algorithm for zero-watermarking in XML data, building on the concept of generating watermarks based on the data's inherent features without altering the original content. They introduced two zero-watermarking methods for XML documents:

- **XSLT-Related Method**: This method embeds extra codes in the XSLT file associated with the XML document. These codes serve as a form of copyright function, ensuring that the watermark remains intact without modifying the original XML content.
- **Functional Dependency-Based Method**: This method uses the functional dependencies (FDs) within the XML file to generate the watermark. By identifying and leveraging the inherent relationships between XML elements, this approach ensures that the watermark is robust against selection, alteration, reorganization, and compression attacks.

The key innovation in Wen et al.'s approach is the use of functional dependencies to generate the watermark. This technique ensures that the watermark is tied to the semantic content of the XML document, making it resilient to various forms of data manipulation that do not alter the fundamental relationships between data elements.

3 Problem Description

3.1 Threat model

The primary threats to XML watermarking stem from attacks that aim to remove or alter the watermark without significantly degrading the data's usability. These attacks are conducted under the constraint that the modified XML must still meet its intended usability requirements, which can be defined through usability queries. The key types of attacks include:

- **Selection Attack:** Removing elements, attributes, or data from the XML document, potentially erasing or corrupting the watermark.
- **Insertion Attack:** Adding new elements, attributes, or data to the XML document to disrupt the watermark without removing existing content.
- **Alteration Attack:** Changing the content of elements or attributes in the XML document, which can alter or destroy the watermark.
- **Zero-Out Attack:** Changing the content of elements to 0 potentially erasing or corrupting the watermark.
- **Reordering Attack:** Adversaries can modify the XML schema, reorder elements, or change the relationships between data elements to disrupt the watermark. For example, converting a flat XML structure into a more nested format can hinder watermark detection if the watermarking method relies on specific structural relationships.
- **Compression Attack:** Compression techniques can be applied to reduce the size of the XML document, which can also lead to the removal of redundant or less critical data elements that might contain the watermark. This type of attack is similar to redundancy removal but focuses more on data size reduction.

Functional dependencies (FDs) represent inherent relationships between data elements in XML documents. Attacks that specifically target FDs aim to disrupt these relationships, thereby compromising the watermark.

3.2 Problem statement

The technique proposed by Wen et al. assumes that functional dependencies will remain unchanged under typical attacks. However, this assumption may not hold in scenarios where attackers deliberately target FDs that are not essential for data usability. As a result, watermarks embedded using

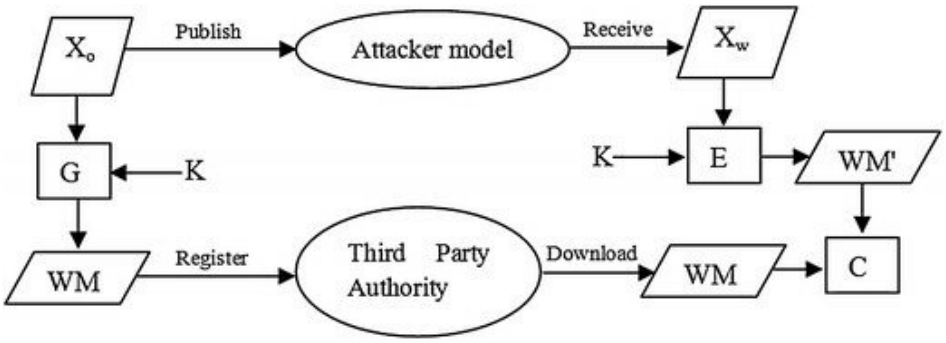


Figure 1: The Zero-watermarking Process

these FDs are vulnerable to deletion, leading to the loss of watermark integrity.

To address this vulnerability, we propose enhancing the robustness of zero-watermarking for XML by incorporating usability queries into the watermark generation process. By filtering functional dependencies based on their relevance to these usability queries, we can ensure that the watermark remains intact even if some FDs are modified. This approach leverages the concept of cover ranges for usability queries to identify and use only those FDs that are critical to the data's usability, thereby increasing the resilience of the watermark against targeted attacks.

```
<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>
```

Figure 2: Book.xml

4 Zero-Watermarking scheme

In this section we discuss the concepts of data usability and zero-watermarking then we present the original watermarking algorithm and our improved version in detail.

4.1 Zero-watermarking

Zero-watermarking is a technique that generates a watermark based on the inherent properties of the data without altering the original content. This distortion-free method is particularly useful for applications where data integrity and imperceptibility are crucial. The general mechanism is described in Fig. 1. Different from most watermarking schemes, instead of embedding the watermark directly into the data, zero-watermarking derives the watermark from the data's structural or semantic features, ensuring that the watermark can be verified without any modifications to the data itself. As shown in Fig. 1, the watermark is generated from the original data X_o with the algorithm G. The feature generated by G should be hard to removed, so after the attacker modifies the original data creating X_w, the watermark can still be extracted using the algorithm E equivalent to G.

The main advantages of zero-watermarking include:

- No Data Modification: Since the original data is not altered, there is no impact on data usability or integrity.
- High Imperceptibility: The watermark is derived from the data rather than embedded into it, making it completely imperceptible.
- Robustness: The watermark is tied to the data's inherent properties, making it resilient to many types of attacks that do not fundamentally alter these properties.

4.2 Original algorithm

Here we describe the original algorithm from [11]. This method leverages that the functional dependencies inherently found in XML data are unique and invariant as most attacks do not alter the semantic information of XML. We define functional dependencies in an XML tree as follows:

Given an XML tree T that conforms to a schema S, a functional dependency is an expression of the form $P1 \rightarrow P2$ where $P1, P2 \in \text{paths}(S)$, T satisfies $P1 \rightarrow P2$. If for any pair of tuples $d1, d2$ in the flat table $FT(T)$, $d1_{P1} = d2_{P1}$ implies $d1_{P2} = d2_{P2}$.

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

- *book*$\backslash$*title* $\rightarrow$ *book*
- *book* $\rightarrow$ *book*$\backslash$*title*
- *book* $\rightarrow$ *book*$\backslash$*rating*

Algorithm 1: genZW

Input: XML data XD, secret key K
Output: Zero-watermark ZW

1 **Function** *genZW(XD, K)*:
2 Convert XD to hierarchical representation R;
3 $FDs \leftarrow$ DiscoverFD(R);
4 $ZW \leftarrow$ GetBinary(FDs, K);
5 **return** ZW;
6 **Function** *GetBinary(FDs, K*:
7 FDs String $S_{FD} \leftarrow$ letters extract($FDs.LHS$, $FDs.RHs$);
8 $N_b \leftarrow$ String2Binary(S_{FD});
9 $WM \leftarrow$ Encode(N_b,K);
10 **return** WM;

Algorithm 2: detectZW

Input: XML data XD', secret key K, original zero-watermark ZW
Output: Similarity score sim

1 **Function** *detectZW(XD', K, ZW)*:
2 Convert XD' to hierarchical representation R';
3 $FDs' \leftarrow$ DiscoverFD(R');
4 $ZW' \leftarrow$ GetBinary(FDs', K);
5 $sim \leftarrow$ Compare(ZW, ZW');
6 **return** sim;
7 **Function** *Compare(ZW, ZW')*:
8 $N \leftarrow$ length of ZW';
9 $sim \leftarrow ZW' \cdot ZW/N$;
10 **return** sim;

- *book\title → book\publisher, book\editor*

Generation algorithm

The generation of the zero-watermark is described in Algorithm 1. They use *DiscoverXFD*, an algorithm proposed by Yu et al. [12], to find the FDs in the XML tree. Since the book.xml schema Weng et al. use for their experiments and the DBLP dataset we later introduce only have a depth of 1, all functional dependencies are *intra-relation* (they involve a single relation). In this case we can replace *DiscoverXFD* with the simpler *DiscoverFD* algorithm from [12], that does not address *inter-relation* dependecies.

At the line 2, before call DiscoveXFD function, XML data set is converted to hierarchical representation which is suitable input for DiscoverFD function. At the line 3, FDs, the output of DiscoverFD have the strings format. At the line 4, the function GetBinary is called to convert the FDs to binary format that can be the ZeroWatermark bits WM. The detail of GetBinary is given from the line 6 to line 10. At the line 9, we encrypt ZeroWatermark bit with the secret key K using the AES symmetric encrytpion algorithm. It is concerned for security issue.

Detection algorithm

Unlike in traditional watermarks, where the detection is the reverse of the embeddig function, in zero-watermarking we detect the watermark by regenerating it from the modified data. We then compare the two watermarks and if the similarity is larger than a threshold $0.5 < T < 1$, we recognize the watermark as ours.

As shown in Algorithm. 2, the detection process also needs to generate the Zero-Watermark bits just like the generating process. However, the input for detection process is XD', which is suspected to violate the copyright and is probably modified or attacked. At the line 2, XD' also need to be converted to hierarchical representation for DiscoverFD function just like the same thing in GenZW. At the lines 3 and 4, DiscoverFD and GetBinary are called again to extract the ZW' from XD'. At the line 5, we use the Compare function to check if there is Zero-Watermark information in XD'. The detail of Compare function is described from line 7 to line 10. The key of Compare function is that the similarity between ZW' and ZW is computed at the line 9. The value of T used in the function Compare is the detection threshold. sim equals to 1 if ZW' is totally equivalent to ZW.

4.3 Modified algorithm - Usability

Wen et al. discovers that the original algorithm is sufficiently resilient against most common attack types for XML watermark, however they make the assumption that attacks do not alter the semantic information of XML, therefore the functional dependencies stay intact. We challenge this assumption as usually a part of the XML schema does not contribute to the data's intended usage. A targeted selection attack poses a threat to the original algorithm, as

Algorithm 3: DiscoverFD

Input: Relation Rp with attributes $a_1, ..., a_n$
QTs – the set of Query Templates
Output: Keys – the set of intra-relation Keys w.r.t. C_p
FDs – the set of intra-relation FDs w.r.t. C_p

Function *Rp, QTs*:
  Generate $c_1, ..., c_n$ from QTs;
  Generate $\Pi_\emptyset, \Pi_{c_1}, ..., \Pi_{c_n}$ from R_p;
  Initialize $Keys = \emptyset, FDs = \emptyset, Queue\ Q = \emptyset, AttributeSet\ A = \emptyset$;
  for $i = 1, ..., n$ **do**
    Q.enqueue($\{c_i\}$)
  end
  while $Q \neq \emptyset$ **do**
    $A = Q$.dequeue();
    if Π_A *does not exist* **then**
      $Ls = candidateLHS(A, FDs)$;
      if *Ls.size == 0* **then**
        continue
      end
      if *Ls.size == 1* **then**
        let $A_1 \in Ls$: $\Pi_A = \Pi_{A-A_1} \cdot \Pi_{A_1}$;
      end
      if *Ls.size >= 2* **then**
        let $A_1, A_2 \in Ls$: $\Pi_A = \Pi_{A_1} \cdot \Pi_{A_2}$;
      end
    end
    if Π_A*.maxGrpSize == 1* **then**
      $Keys$.add(A);
      continue;
    end
    foreach $A_L \in Ls$ **do**
      let $r = A - A_L$; `// A_L is the LHS, r is the RHS`
      if $\Pi_{A_L} == \Pi_A$ **then**
        FDs.add($A_L \rightarrow r$);
      end
    end
    let a_k be the last attribute in A;
    for $i = k + 1, ..., n$ **do**
      $A' = A \cup \{a_i\}$;
      if *there is no* $K \in Keys$, *such that* $K \subseteq A'$ **then**
        Q.enqueue(A');
      end
    end
  end
Function *candidateLHS(A, FDs)*:
  Initialize $Ls = \emptyset$, the set of LHSs to be returned;
  if ***not*** FDs **then**
    Ls.add(A_L);
  end
  foreach $a \in A$ **do**
    let $A_L = A - \{a\}$;
    foreach $L \rightarrow r \in FDs$ **do**
      if $a == r$ *and* $A_L \subseteq L$ **then**
        continue;
      end
      else if $A \subseteq L$ **then**
        continue;
      end
      else
        Ls.add(A_L);
      end
    end
  end

the adversary could delete a large portion of data elements that are dispensable with regards to the data's intended use. This would remove all functional dependencies including these schema elements, severely damaging the watermark. In this section we discuss the definition of usability and our modified version of DiscoverFD that selects only the usable FD-s.

Zhou et al. defines XML data usability in the form of query templates. We take on these definitions from [13]:

Query: A query is a string "$a_1[\text{sat}_1]/a_2[\text{sat}_2]/\ldots/a_k[\text{sat}_k]$", where "$a_1/a_2/\ldots/a_k$" is a path. $[\text{sat}_i]$ is a selection criterion in the form $[p_{i1} = \text{value}_{i1}, \ldots, p_{ik} = \text{value}_{ik}]$, where each "$p_{ij}$" is a sub-path and each value_{ij} is a character string. The results to the query are the nodes in $\{a_1/\ldots/a_k\}$ that satisfy all the sat_i criteria.

For example, "book[title = 'DB design']/author" is a query to retrieve the authors of the book titled "DB design".

Query Template: A query template is in the form "$a_1[\text{sat}'_1]/a_2[\text{sat}'_2]/\ldots/a_k[\text{sat}'_k]$", where $[\text{sat}'_i] = [p_{i1}, \ldots, p_{ik}]$ (without comparison operation). We say a query fits into a query template if they are equal after we remove all the comparison operations of the query.

For example, "book[title = 'DB design']/publisher" fits in query template "book[title]/publisher".

Cover Range: For a query template $q = a_1[\text{sat}'_1]/\ldots/a_k[\text{sat}'_k]$, where $[\text{sat}'_i] = [p_{i1}, \ldots, p_{ik}]$, its cover range $C(q)$ is the set of paths that comprise $a_1/\ldots/a_k$ and all $a_1/\ldots/a_i/p_{ij}$.

For example, in book.xml, the cover range of "book[author, editor]/ title" is "book/title", "book/ author ", "book/editor ".

In simple terms, if the book.xml dataset's intended usage is to find the rating of a book by its title, and find all books from a given author, the query templates *"book[title]/rating"* and *"book[author]/title"* describe the usability and the cover range of these templates will collect all important elements which are safe from getting targeted by an attacker as it would destroy the usability. In our modified algorithm, we generate the cover ranges for user defined query templates and select only the FD-s related to elements in the cover ranges.

Algorithm 3 describes the DiscoverFD function from [12] used in the generation procedure. We modified it to only extract functional dependencies from the attributes that appear in the cover range of the query templates. The main data structure used is *attribute partition*. An attribute partition of an attribute set X (ΠX) is a set of partition groups, where each group contains all tuples sharing the same values at X. The DiscoverFD algorithm aims to identify all intra-relation functional dependencies. It operates by traversing a lattice structure and constructing attribute partitions to determine Keys and satisfied minimal FDs. The lattice traversal is managed using a queue, which processes nodes bottom-up. For each node, the algorithm checks if the associated partition indicates a Key, where a Key is an attribute set whose partition groups contain exactly one tuple. Additionally, it

detects satisfied FDs by comparing partitions. The algorithm also combines partitions of smaller attribute sets to produce new partitions. Our additions are described in line 1-2 where we generate the cover ranges of the user-defined query templates, returning a set of attributes $c_1, ..., c_n$. We then only generate the partitions for these attributes in line 3. We also added the lines 44-46. Without this Ls and FDs could not populate as they depend on each other and both is initialized as empty sets.

5 Experimental Setup and Results

We have implemented both algorithms and designed a set of simulated attacks to test the resilience of the two methods. All of the simulation procedures are executed on an Intel Core i7 CPU running at 3.3 GHz with 16 GB physical memory. The implementation is published on GitHub [10].

To conduct our experiments, we use the DBLP computer science publication database [2]. To simplify our study, we only consider the *'inproceedings'* elements. Zhou et al. describes the dataset's usability with these applications:

- find the authors of a given publication;
- find the publications of a given author;
- find the publications of a particular conference or journal.

After initial experiments, we decided to add two more applications as using only the aforementioned three, the watermark capacity is insufficient against any types of attacks.

- find the year a given article was published in;
- find the publications of a given book.

These are converted into the following query templates:

- *"Root/inproceedings[title]/author*
- *"Root/inproceedings[author]*
- *"Root/inproceedings[conference]/title*
- *"Root/inproceedings[title]/year*
- *"Root/inproceedings[booktitle]/title*

Here, we focus on the robustness of the two watermarking methods. We execute five type of attacks on the dataset and measure the detectability of the watermarks. We start the attacks with selection to set a baseline for our modification against the more common attacks discussed in [11]. We then execute zero-out, targeted selection, targeted-zero out and a selection attack only deleting one specific attribute of the schema. We simulate the first four attacks with random sampling, therefore all experiments were run multiple times and the average of the results is presented. Fig. 3 shows the similarity between the original and detected watermarks over the amount of elements attacked ranging 10-80%.

In a selection attack the adversary selects part of the data and discards it with hopes that the watermark is destroyed in the process. It is shown in Fig. 3a that the original model performs better than ours, due to it's significantly higher watermark capacity. Even if a large part of the data is removed, some functional dependencies can be recovered.

The zero-out attack, commonly used against relational data, instead of discarding the selected part of the data, sets its values to "0". Fig 3b shows that the original algorithm is extremely fragile against any zero-outs as unifying the values in more than 2 elements introduces extra functional dependencies between previously unique attributes detected as keys. Our algorithm shows more resilience against this attack, since the cover range does not contain key attributes.

The next two attacks are targeted in terms of data usability. Here the adversary instead of randomly selecting, tries to preserve the data's functionality by only attacking attributes that are not useful. The targeted selection attack again selects part of the inproceedings elements, but instead of completely discarding them, only deletes the attributes outside of the cover range of their query templates. Fig 3c shows that if the attacker follows the data's intended use, our algorithm is extremely robust against the selection. The original algorithm's performance is similar to the one against standard selection.

The targeted zero-out attack follows the same principle as the previous one, but with changing the attribute values to "0" instead of deleting them. It is shown again in Fig 3d that the method in Wen et al. cannot address zero-outs due to the large number of FDs created by it.

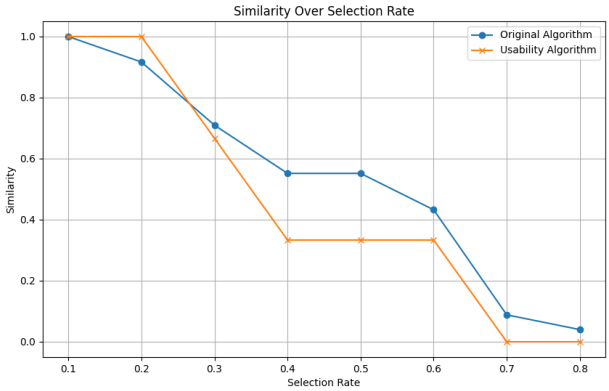
The last attack is single attribute selection, where the attacker completely deletes the least significant attribute from the schema in terms of usability. Fig 4 shows the detection rates for the two algorithms for each attribute deleted. We did not include the attributes *"title"* and *"author"* as they are arguably indispensable. These results give insight on what attributes contribute to a large part of the FDs.

6 Conclusion and Future Work

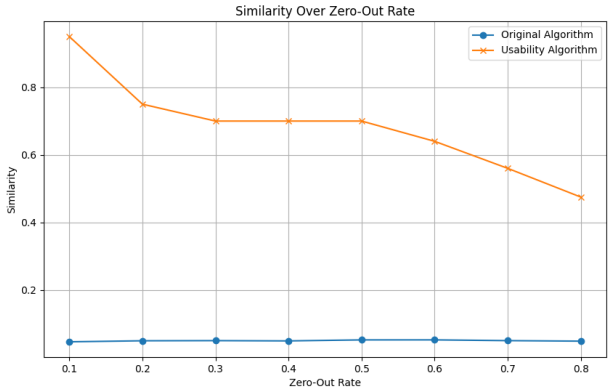
In this paper we introduced context-based targeted attacks against XML watermarks and verified that the integration of Usability Queries into the Zero-Watermarking method using functional dependencies proposed by Wen et al. increases robustness against them. Various attack types such as the Zero-Out Attack, have been included in our experiments. Since the pseudocodes describing both algorithms (genZW and discoverFD) we implement leave much room for interpretation and the dataset used in Wen et al. is not public, we could not manage to reach the baseline set by their experiments. We do however show, that Zero-Out attacks pose immense threat to FD based algorithms as they generate a large amount of extra dependencies.

Responsible Research

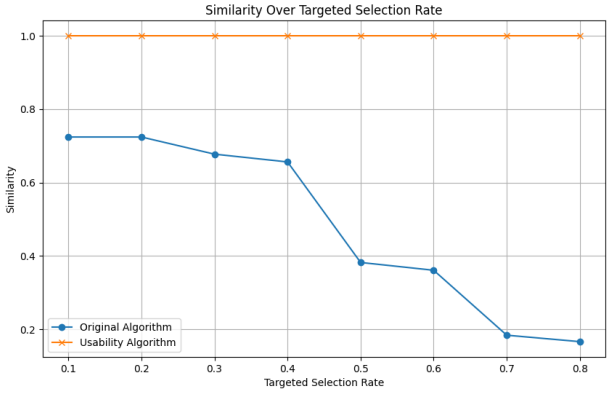
Our research on enhancing XML zero-watermarking techniques adheres to the highest ethical standards and prioritizes transparency, reproducibility, and data privacy. By utilizing the publicly available and trustworthy DBLP dataset, we ensure the integrity of our experiments. We provide comprehensive documentation of our methods, and our source code will be open-sourced on GitHub to facilitate replication and further research. This commitment to openness supports the principles of scientific integrity outlined in the Netherlands Code of Conduct for Research Integrity. By integrating usability queries into the watermarking process, we aim to en-



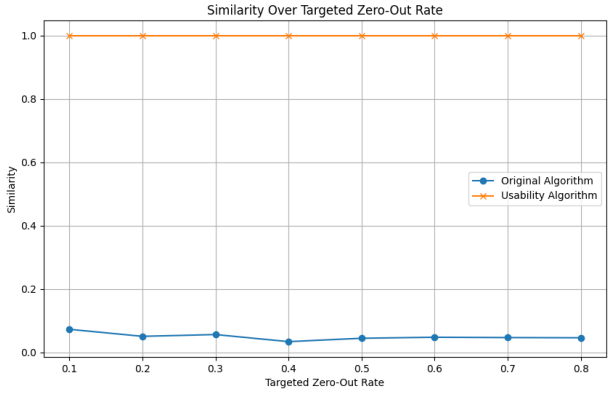
(a) Selection Attack



(b) Zero-Out Attack



(c) Targeted Selection Attack



(d) Targeted Zero-Out Attack

Figure 3: Robustness Analysis For Selection And Zero-out Attacks

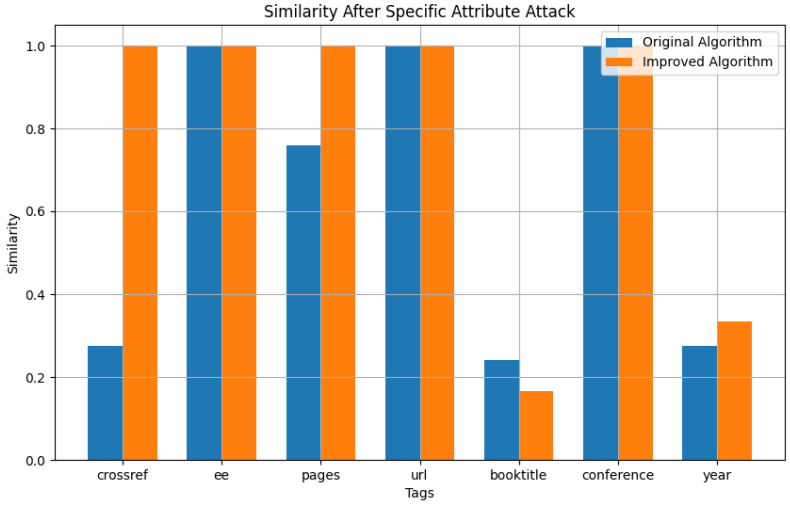


Figure 4: The Single Attribute Selection Attack

hance the robustness of functional dependency-based watermarking while advocating for the responsible and ethical use of these technologies.

Future Work

Future research should focus on utilizing different datasets with the completed DiscoverXFD algorithm detecting also inter-relation dependencies to evaluate the robustness and effectiveness of the watermarking approach across various types of XML data. An optimized version of the algorithm was proposed in [7], implementing this method will prove effective on large-scale datasets. Additionally, it is necessary to explore and identify more targeted attack types to challenge the integrity of the watermarked data. Understanding these attack vectors will be crucial in enhancing the resilience of the watermarking technique.

Developing a hybrid watermarking method that balances usability and watermark capacity is also a critical area for future work. Such a method should selectively use functional dependencies that relate to the cover ranges of user-defined usability queries while maintaining a larger portion of the watermark capacity. This approach aims to increase the robustness of the watermark against semantic attacks without compromising the data's usability. Furthermore, it is important to test the impact of different adversary query templates on the effectiveness of the watermarking method. By simulating various adversarial scenarios, researchers can fine-tune the approach to ensure it remains secure and functional under a wide range of potential threats.

References

[1] Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, 2002.

[2] Michael Ley. Dblp computer science bibliography. http://dblp.uni-trier.de/, 1993–2023. Accessed: 2024-06-19.

[3] Yunhua Li, Shuguo Guo, and Sushil Jajodia. Tamper detection and localization for categorical data using fragile watermarks. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 135–146, 2004.

[4] Wilfred Ng and Hoi-Lam Lau. Effective approaches for watermarking xml data. In *Proceedings of the 2005 International Conference on Information and Knowledge Management*, pages 429–436, 2005.

[5] A. Soltani Panah, R. Van Schyndel, T. Sellis, and E. Bertino. On the properties of non-media digital watermarking: A review of state of the art techniques. *IEEE Access*, 4:2670–2704, 2016.

[6] S. Rani and R. Halder. Comparative analysis of relational database watermarking techniques: An empirical study. *IEEE Access*, 10:27970–27989, 2022.

[7] Hui Shi, Tetsuya Amagasa, and Hiroyuki Kitagawa. Fast detection of functional dependencies in xml data. In *Database and XML Technologies*, volume 6309 of *Lecture Notes in Computer Science*, pages 113–127. Springer Berlin Heidelberg, 2010.

[8] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Resilient information hiding for abstract semi-structures. In Ton Kalker, Ingemar Cox, and Yong Man Ro, editors, *Digital Watermarking*, volume 2939 of *Lecture Notes in Computer Science*, pages 141–153. Springer Berlin Heidelberg, 2004.

[9] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Rights protection for relational data. *IEEE Transactions on Knowledge and Data Engineering*, 16(12):1509–1525, 2004.

[10] Benedek Szekacs. Query-based xml watermarking. https://github.com/bszekacs22/Query-Based-XML-Watermarking, 2024. Accessed: 2024-06-19.

[11] Zhong Wen, Xiangliang Wang, and Yongjian Li. Zero-watermarking for xml data based on functional dependencies. *Journal of Real-Time Image Processing*, 13(2):313–324, 2016.

[12] Cong Yu and H.V. Jagadish. Efficient discovery of xml data redundancies. In *VLDB '06 Proceedings of the 32nd international conference on very large data bases*, pages 103–114. VLDB Endowment, 2006.

[13] Xuan Zhou, HweeHwa Pang, and Kian-Lee Tan. Query-based watermarking for xml data. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 437–448, 2007.