Graph Database Watermarking Using Pseudo-Nodes

Tsvetomir Hristov
Cyber Security Group, Delft University of Technology
The Netherlands
T.Hristov@student.tudelft.nl

Devriş İşler
IMDEA Networks Institute & UC3M
Spain
devris.isler@imdea.org

Nikolaos Laoutaris
IMDEA Networks Institute
Spain
nikolaos.laoutaris@imdea.org

Zekeriya Erkin
Cyber Security Group, Delft University of Technology
The Netherlands
Z.Erkin@tudelft.nl

ABSTRACT

Watermarking is used as proof of ownership for various data types such as images, videos, software, machine learning models, and databases. Datasets are crucial for data driven decision making using Machine Learning for tasks like prediction, recommendation, classification, and anomaly detection. Hence, it is not surprising that entire databases are being sold in data marketplaces. Protecting ownership rights upon such databases is, therefore, becoming increasingly important. Watermarking for relational databases has been an active field of research since 2002. However, how to watermark non-relational databases involving complex data types has largely remained understudied. In this paper we revise previously proposed techniques for non-relational database watermarking and introduce an improved technique for graph database watermarking inspired by Zhuang et al. [28]. Our technique employs randomization to generate a watermark in an efficient manner that avoids the computational complex genetic algorithm optimization of Zhuang et al. We evaluated our technique in terms of performance, usability, security, and robustness by implementing it as a proof-of-concept. Our results showed that our technique is efficient, secure and robust against guessing and deletion attacks.

CCS CONCEPTS

• **Social and professional topics** → **Computing / technology policy**; • **Intellectual property** → Digital rights management .

KEYWORDS

Graph database, watermarking, data ownership, data economy

ACM Reference Format:
Tsvetomir Hristov, Devriş İşler, Nikolaos Laoutaris, and Zekeriya Erkin. 2023. Graph Database Watermarking Using Pseudo-Nodes. In *Second ACM Data Economy Workshop (DEC '23), June 18, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3600046.3600049

DEC '23, June 18, 2023, Seattle, WA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0846-6/23/06.
https://doi.org/10.1145/3600046.3600049

1 INTRODUCTION

With the rapid growth of data driven technologies, data is becoming a vital ingredient for the modern economy. Data marketplaces (DMs) enable data buyers to buy data for important tasks, such as training machine learning model, providing better service for costumers, etc. One of the main challenges faced by DMs is how to protect ownership rights upon data against unauthorized distribution, copying (pirating) and counterfeiting.

Watermarking [17] is a well-established and often-used technique for protecting and validating intellectual property rights. Watermarking embeds information into data by using secrets (e.g., private key, a sequence of bits) only known by the owner. The secret knowledge inside watermarked data is later used as proof for claiming ownership. Watermarking for proof of ownership has been extensively studied for various types of data, including audio, video [5], images [4], software [9, 26], databases [3, 14], different dataset types [6, 23, 29], and (deep) neural networks [2, 27]. Despite the fact that non-relational data is steadily growing in popularity due to their lack of schemas adaptivity [7, 11], watermarking for non-relational databases is understudied. To store such enormous and useful data consisting of complex data types and structure, NoSQL databases are introduced to overcome the limitations of traditional databases.

In this paper, we investigate the existing watermarking approaches and focus on a work by Zhuang et al. [28] in which the authors embed a watermark in a MongoDB database using connected graphs. Even though the approach in [28] is elegant, we propose an improved version that does not require computationally complex genetic algorithms while still remaining secure and robust against *guess* and *deletion* attacks. Furthermore, our technique is capable of detecting who leaked the database in addition to proving ownership.

More precisely, our technique as proposed by Zhuang et al. [28] generates pseudo documents from a graph database, in which documents in the collection are represented as a graph. The generation of pseudo documents is based on random pre-selected collections/groups from the database. Then each pseudo document is marked using a high entropy secret which is later used as a part of the watermarking secret, in addition to pseudo documents identifiers (e.g., primary keys). To insert the watermarked pseudo document into the original database, the original database is partitioned in groups, and each watermarked pseudo document is linked to one of the groups. When an owner suspects a copy of its database being distributed without its authorization, it can prove its ownership on the suspected database by running watermark extraction algorithm. It first identifies the pseudo documents and then verifies if identified possible pseudo document(s) are marked by the private key.

We further implemented our technique with different parameter settings using a real-world dataset about UK Companies [21] supported by Neo4j to evaluate: 1) how much pseudo documents change the original database which is ∼ 4.9%; 2) watermark generation time which takes around 272 seconds on average, 3) security and robustness against deletion attack.

Our contributions: Our first contribution is the development of a new watermarking scheme for graph database by improving the previous state of the art work due to Zhuang et al. [28] by simplifying it using randomization. We later implemented our scheme in open source to evaluate its performance using a real world dataset with different parameter settings. Finally, we analyzed the security and robustness of our scheme against *guessing* and *deletion* attacks and showed that it is resilient to such attacks.

2 RELATED WORK

In this section, we briefly introduce relational database watermarking and NoSQL database watermarking.

Relational database watermarking. The first known technique was proposed by Agrawal et al. [3] in 2002. Later, other techniques are introduced for numerical databases by considering the statistics of numeric values [20, 24]. Watermarking techniques introduce some sort of distortion to the original data which may not be ideal for some cases. To avoid distortion, *distortion-free database watermarking* schemes have also been proposed: [10] is a distortion-free technique that introduces fake tuples or columns in the original database.

NoSQL database watermarking. To the best of our knowledge, there are few watermarking schemes introduced for NoSQL databases. Khandu et al. [19] introduced a NoSQL database watermark scheme where each new entry is watermarked before being added to the database. Than et al. [25] proposed a watermarking technique for NoSQL databases that modifies the structure of the key-value attributes for hiding information as proof of ownership.

Zhuang et al. [28] watermarks a MongoDB database using connected graphs as illustrated in Figure 1. The documents from the database are used to build a graph, where references between documents are represented as edges, and documents as nodes. Then, pseudo documents are created using a genetic algorithm , a watermark is embedded into the pseudo nodes and they are added into the database. Finally the documents are connected into a k-connected graph. To detect the watermark, the algorithm finds the largest k-connected graph and checks the watermark for those documents.

Considering the work by Zhuang et al.[28], we summarize its shortcomings in the following:

Limitations of genetic algorithm. Genetic algorithms are designed to tackle complex problems in optimization and search [15] by optimizing a fitness function. Zhuang et al. deploy genetic algorithms as a means of pseudo-document generation. Unfortunately, how this part is implemented is not defined in the original work, which makes their scheme impossible to reproduce. Furthermore, an attacker can easily distinguish pseudo documents from real/original documents by observing the distance to local/global maxima or minima.

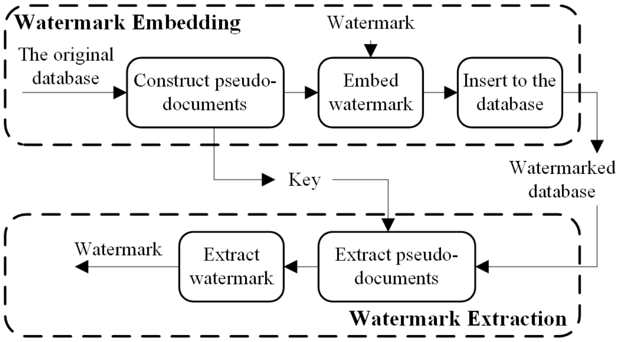

Figure 1: Overview of Zhuang et al. [28].

Graph creation. Zhuang et al. group pseudo documents which are connected to form a k-connected graph. While the approach is novel, an attacker can detect the pseudo documents used for watermarking by analyzing the graph generated from connections of documents, e.g., pseudo documents in a group will be connected to each other. Thus, the attacker can successfully remove the watermark.

3 GRAPH DATABASE WATERMARKING

3.1 Background

SQL and NoSQL Databases: SQL and NoSQL databases vary by terminology: an SQL database consists of tables, which contain rows, also called records, and columns. On the other side, NoSQL databases consist of collections containing documents that store information into key/value pairs, as explained in Figure 2.

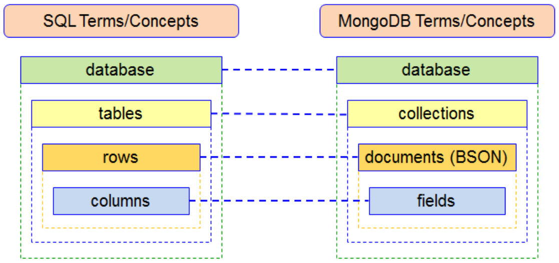

Figure 2: Sketch showing the difference in terminology between SQL and NoSQL database, published by Studio 3T, a GUI tool for Mongo [1].

A typical SQL database stores all its information in tables, where each table has a well-defined schema, and each entry must abide by this scheme. A NoSQL database does not enforce a scheme, which makes watermarking harder, as there is no guarantee that a document follows a specific data model. Furthermore, NoSQL databases vary by how they store relations between documents. A typical SQL database uses a table to store relations between records, whereas, for NoSQL databases, each type tackles the problem in their own way. For example, document databases offer multiple solutions: nesting/embedding, and referencing, while graph databases use edges. Each edge between two nodes, or documents, is directed and can store additional data about the relation. We use nodes and documents interchangeably throughout the paper. In Table 1 we present our method notations used throughout the paper.

Table 1: Notations.

Notation	Description
$\mathcal{D}$	The original database
$\mathcal{D}_w$	The watermarked database
D_i	i^{th} record of the original database
D_w^i	i^{th} record of the watermarked database
$\mathcal{K}$	A private key used for the watermarking process
$\mathcal{P}$	A pseudo document
$G^{\mathcal{D}}$	A group of documents from the database
$G^{\mathcal{P}}$	A group of pseudo documents
$ID_G^{\mathcal{P}}$	The ids of watermarked documents inside the database

3.2 Overview

Our new private key-based graph database watermarking technique inspired by Zhuang et al. [28] is *blind* in the sense that it does not require the original database, *robust*, and *secure* against **deletion** and **guessing** attacks.

As illustrated by Figure 3, our watermarking scheme consists of two main algorithms: 1) watermark embedding, and 2) watermark extraction. The embedding algorithm watermarks a graph database using a high entropy secret. By calling the watermark embedding algorithm, the owner creates a watermarked version of its database which allows it to prove its ownership. Watermark extraction detects if a suspected database is watermarked by the owner using the secrets produced by embedding. If it outputs *accept/verified*, this evidence would prove that the owner can claim ownership of the watermark and thus the database.

3.3 Embedding

Embedding, shown in Algorithm 1, takes two inputs: an original graph database $\mathcal{D}$ and a high entropy private key $\mathcal{K}$ generated by a key generation function as $\mathcal{K} \leftarrow KeyGen(1^{\lambda})$. Note that the key generation algorithm can be computed offline by the owner.

The algorithm steps are as follows:
1) Generates groups of documents $G^{\mathcal{D}}$ from $\mathcal{D}$ where each document in $\mathcal{D}$ belongs to only one group as $G^{\mathcal{D}} \leftarrow GroupGen(\mathcal{D})$.
2) Generates a group of pseudo documents $G^{\mathcal{P}}$ using the original dataset $\mathcal{D}$ and group information of the dataset $G^{\mathcal{D}}$ as $G^{\mathcal{P}} \leftarrow PseudoGen(\mathcal{D}, G)$. **Remark:** Pseudo documents are synthetic documents that are generated based on the information of the field they belong to. We explain the pseudo document generation later.
3) Each pseudo document in $G^{\mathcal{P}}$ is marked as $G_w^{\mathcal{P}} \leftarrow Mark(G^{\mathcal{P}}, \mathcal{K})$ by using a private key $\mathcal{K}$ to generate a marked version $\mathcal{G}_w$.
4) After generating $\mathcal{G}_w$, each marked pseudo documents must be inserted to $\mathcal{D}$ to create a watermarked database $\mathcal{D}_w$. To do so, each watermarked pseudo document in $\mathcal{G}_w$ is linked to a group.
5) Returns the watermarked database $\mathcal{D}_w$, the private key $\mathcal{K}$, and the list of identifiers of pseudo documents $ID_G^{\mathcal{P}}$.

Algorithm 1 Watermark Embedding

```
Input: $\mathcal{D}, \mathcal{K}$
Output: $\mathcal{D}_w$
$G^{\mathcal{D}} \leftarrow GroupGen(\mathcal{D})$
$G^{\mathcal{P}} \leftarrow PseudoGen(\mathcal{D}, G)$
$G_w^{\mathcal{P}} \leftarrow Mark(G^{\mathcal{P}}, \mathcal{K})$
$\mathcal{D}_w := []$
for $D_i \in G^{\mathcal{D}}$ do
    $\mathcal{D}_w.add(Link(D_i, G_w^{\mathcal{P}}))$
end for
return $\{\mathcal{D}_w, \mathcal{K}, ID_G^{\mathcal{P}}\}$
```

After embedding, the owner stores the private key $\mathcal{K}$, and the list of identifiers of pseudo documents $ID_G^{\mathcal{P}}$ as a watermarking secret in secure storage to use later for extraction. Note that the secret is used as proof of ownership; thus, the owner has to protect it.

Group Generation: *GroupGen* algorithm partitions the nodes in each collection of $\mathcal{D}$ into groups. For this purpose, an algorithm is used to find a random list of integers whose sum adds up to a number. The algorithm is then given the *min* and *max* parameters as the range and the number of nodes, which need to be watermarked, as the total sum. The output is then used to partition the nodes into groups. For example, if the algorithm is run with the range $(3, 5)$ and a target sum of 15, a viable solution is $[3, 4, 3, 5]$. This means the first group will have 3 nodes (documents), the second 4 documents, and so on. Determining optimal *min* and *max* requires further investigation as we leave it as a future work.

Pseudo Documents Generation. A set of pseudo documents are generated by an algorithm *PseudoGen* based on the original database $\mathcal{D}$ and group/collection information of $\mathcal{D}$. *PseudoGen* creates these pseudo documents by "borrowing" fields from other real/original documents in $\mathcal{D}$ where the size of pseudo documents is determined (e.g., randomly). First, for each pseudo document, it randomly chooses a set of fields from some randomly chosen "donor" documents in $\mathcal{D}$ and create the pseudo documents.

To determine which fields are "borrowed" a schema analysis is manually performed beforehand, which is explained in more detail in the next section. Note that different techniques can be deployed (e.g., synthetic data generation [12]) for the creation of pseudo documents. Investigation of such techniques and their impact on our watermarking scheme are left as a future work as it requires further analysis.

Marking. To mark a document, a simple algorithm is used to calculate the watermark. *Mark* function overwrites a numerical field inside the pseudo document to carry the watermark using the private key $\mathcal{K}$. For each numerical field[1], a modulo operation is computed as

[1] We only watermark the numerical values. However, watermarking text values is possible although it requires deep knowledge of the context that the texts are used [28].

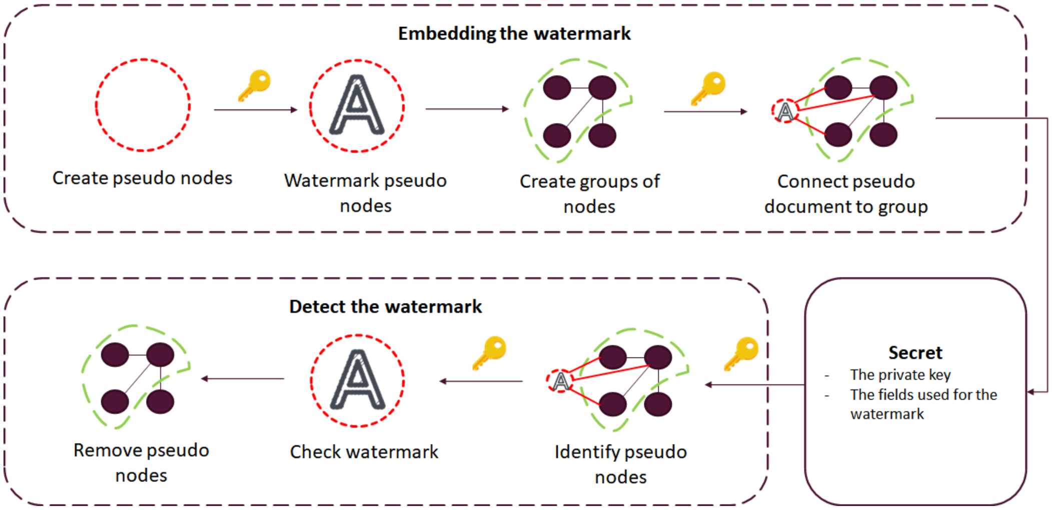

Figure 3: Overview of our graph database watermarking.

$new = H(fields||\mathcal{K}) \mod fieldmax$ where $fieldmax$ represents the highest number for the fields in the original database and H is a collision resistant hash function. Later, the original field value is replaced with new.

Linking and adding pseudo documents: After generating a watermarked psuedo doc and an groups of original documents that they will be inserted into, the challenge is to link pseudo documents to the groups. *Link* algorithm is computed on the pseudo documents. Linking algorithm treats each pseudo document as if it is a new entry inserting to the database.

As the primary key can be used to hint about the location of the pseudo documents, several ways to add a watermark inside the database can be considered based on the way the database stores the documents. For example, the primary key for a MongoDB database is a random string; however, the primary key for a Neo4j database is an integer. This research focuses on embedding documents into a Neo4j database; however, similar techniques can be used for other NoSQL databases.

Unfortunately, Noe4j has a limitation, where the primary key of a node cannot be changed. This is why for the purpose of this research, the watermarked nodes are simply added without any manipulation of the primary key. To recall the pseudo documents, their primary keys are also stored as a part of the watermarking secret.

3.4 Extracting

In Algorithm 2, we provided our watermarking extraction as a pseudocode. The owner suspects a graph database $\mathcal{D}_w$' that might belong to it. Inputs of the watermark extraction are a suspected database $\mathcal{D}_w$', the private key $\mathcal{K}$, and the identifier list of marked pseudo documents $ID_G^{\mathcal{P}}$ where the owner already possesses $\mathcal{K}$ and $ID_G^{\mathcal{P}}$. For watermark extraction, the following steps are taken:
1) Retrieve each document with identifiers in $ID_G^{\mathcal{P}}$.
2) Check if the retrieved documents are marked by $\mathcal{K}$ by running *MarkCheck* algorithm. *MarkCheck* compares the hashes of the original fields in $ID_G^{\mathcal{P}}$ and the fields $fields'$ retrieved from $\mathcal{D}'_w$ and returns accept if they are equal as $H(fields||\mathcal{K}) \mod fieldmax = H(fields'||\mathcal{K}) \mod fieldmax$. **Remark.** Note that as in previous (non-)relational database watermarking schemes, a threshold can be set to decide if a document is marked as well as at least how many documents shall be retrieved. For our technique, retrieving one document for checking the mark is enough although requiring a higher number of documents for marking can decrease the false positive. In our work, we do not investigate such optimization.

Algorithm 2 Watermark Extraction

```
Input: $\mathcal{D}_w$', $\mathcal{K}$, $ID_G^{\mathcal{P}}$
Output: $Accept/Reject$
$G_w^{\mathcal{P}} \leftarrow Find(\mathcal{D}'_w, ID_G^{\mathcal{P}})$
$result = MarkCheck(G_w^{\mathcal{P}}, \mathcal{K})$
return $result$
```

4 EXPERIMENTAL SETUP AND RESULTS

4.1 Experimental Setup

Our experimental results are produced on a standard laptop machine with AMD Ryzen 93900XT 12-Core CPU 3.80 GHz Processor, 16GB of RAM, and a 64-bit Windows 11 operating system. We implemented our watermarking scheme as a proof of concept programmed in Python and is publicly available [13]. The tests are run inside a virtual environment [22], to ensure that the different test setups do not interfere with each other. The tests were repeated 100 times and then averaged.

Dataset. We use a real world dataset UK Companies [21] which is one of the default example datasets by Neo4j. The dataset contains information about 35000 public companies in the United Kingdom, their properties, owners, and political donations. The database has four labels: 1) *Person* with 23606 entries; 2) *Company* with 26226 entries; 3) *Recipient* with 9 entries; and 4) *Property* with 4728 entries.

Most documents in the UK Company database share an edge with a document of type *Company*. Thus, every node of type *Person*, *Property*, or *Recipient* is watermarked by a node of type *Company*. In turn, watermarking nodes of type *Company* are linked by pseudo nodes from the other three types. This ensures that every node inside the database can be watermarked; however, for simplicity, we only consider nodes/documents with type *Property* in our evaluation.

Since graph databases inherit their schematic nature from NoSQL databases, there is no defined structure for each document type. However, a lot of the documents still share the same fields. An example of a *Company* document can be seen below:

```
{
    "companyNumber": "04179322",
    "name": "CURO TRANSATLANTIC LIMITED",
    "mortgagesOutstanding": 1,
    "countryOfOrigin": "United Kingdom",
    "incorporationDate": "2001-03-14",
    "category": "Private Limited Company",
    "SIC": "64999 - Financial intermediation
        not elsewhere classified",
    "status": "Active"
}
```

Figure 4: Example of a *Company* document from the UK Company dataset.

To evaluate our scheme, we used two types of metrics: 1) performance based on timing; and 2) usability based on the number of pseudo documents added to the dataset using different parameter settings.

The first metric records the time for the watermarking algorithm to embed a watermark and the amount of documents introduced. Log files are manual analyzed for the average time for each group to be watermarked. The speed of the embedding algorithm is deduced using the following formulas:

$$S_D = \frac{N_d}{t} \quad \text{and} \quad S_G = \frac{N_g}{t}\ , \tag{1}$$

where t is the time in seconds it took for the algorithm to finish, N_d is the number of documents in the database, and N_g is the number of groups that were formed by the algorithm. The formula measures the number of documents watermarked per second - S_D; and the number of groups watermarked per second - S_G. Since the algorithm randomly partitions the documents into groups and the heaviest computation is performed per group, the speed per group is considered a more important metric for performance.

Additional testing is conducted on the impact of parameters on the general performance of the algorithm. Parameters such as minimum group size *min* (which is selected from $(5, 10)$), maximum group size *max* (which is selected from $(10, 200)$), and the size of watermarked documents (which the size of pseudo documents) are considered to have a high impact on the performance.

The usability of the watermark is evaluated by several key factors: the number of pseudo documents added, the average, mean, and median of certain numerical values, the number of edges per node, etc. To ensure the usability of the database after the watermark, manual analyses are performed on predefined values, looking for changes introduced by the watermark.

4.2 Results

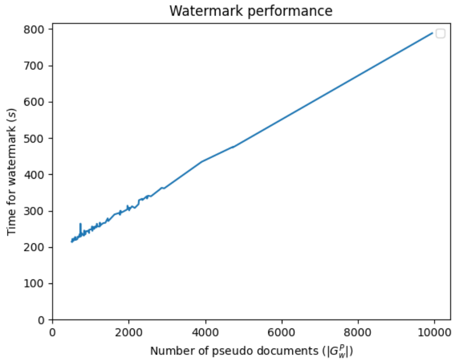

Figure 5: Performance of the algorithm.

Performance. The algorithm's performance was measured in the number of seconds the algorithm took to watermark all groups. Because the number of groups is not fixed, we noted the number of groups that needed to be watermarked and the time it took to watermark them.

Figure 5 shows that the amount of groups that need to be watermarked directly impacts the algorithm's performance. As shown, the algorithm follows a linear time complexity, indicating that the algorithm can be scaled for large databases without compromising performance. Additional calculations show that our scheme watermarks 5.28 groups per second on average without parallelization/multithreading.

Usability. As all previous graph database watermarking schemes, our technique also introduces distortion to the original database by adding pseudo documents. To evaluate the amount of distortion in the original database after embedding, we measured the number of pseudo documents inserted by setting various minimum and maximum group sizes. Figure 6 shows that when the difference between maximum and minimum group sizes increases, the number of pseudo documents introduced decreases; thus, the amount of distortion decreases. Consequently, when we have small group sizes leading to a small difference between *min* and *max*, a higher number of pseudo documents are generated. Note that the more pseudo documents the embedding introduces, the more distortion it introduces while it increases the robustness.

5 SECURITY AND ROBUSTNESS ANALYSIS

In our security analysis (sketch), we assume that a probabilistic polynomial time (PPT) attacker 1) knows how our algorithms work, following a no-security-by-obscurity principle [18], and 2) cannot break the security of underlying algorithms such as hash function, pseudorandom generator.

We assume that all the underlying algorithms used as building blocks are secure by their definitions [16]. The attacker has full

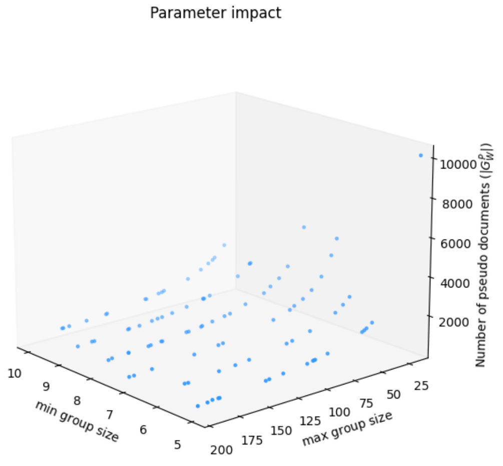

Figure 6: Impact of parameters on the amount of watermarked documents.

access to the public (watermarked) database, which might also be just a part of the full database. The attacker also has access *only* to *one* watermarked copy of the database and does not have access to the original database or the key for the watermark. An attacker cannot check the watermark, as in an oracle attack, to check if the watermark is still valid [8]. Finally, it is assumed that an attacker does not modify the primary keys.

5.1 Guessing Attack

In the guessing attack, a PPT adversary tries to guess the watermark by exploiting the private key used to embed and determining the pseudo documents (i.e., nodes introduced to the graph database after embedding) introduced by the watermarking scheme. By doing so, the attacker can perform other attacks such as watermark removal. Assuming that the hash function is collision-resistant, $\mathcal{K}$ is random, the pseudo documents generations are random, and the link is not predictable to the attacker, the success rate of the guessing attack is negligible. Since the pseudo documents are indistinguishable from the real documents, the attacker cannot determine the pseudo documents to remove them from the watermarked database in order to make the extraction algorithm fail (i.e., returning reject while the database was indeed watermarked by the owner).

5.2 Deletion attack

For the deletion attack, an attacker randomly picks some documents and deletes them from the watermarked database. The attacker tries to delete the documents hoping that the watermark extraction will fail (i.e., returning reject). We simulated such an attack and then run the watermark extraction algorithm to determine if the attack was successful. For the percentage of successful detection attacks, the following formula is used:

$$\%_{suc} = \sum_{A_u \in A} \frac{A_u}{A}\ , \qquad (2)$$

where A_u denotes the number of unsuccessful attacks and A denotes the number of attacks performed. The robustness of the watermark is tested with a deletion attack. First, a script, acting as a malicious actor, randomly picks nodes and deletes them. After that, a verification script is run to determine the amount of watermarked nodes that were deleted or not detected. This sequence continues until the verification script does not detect the watermark anymore.

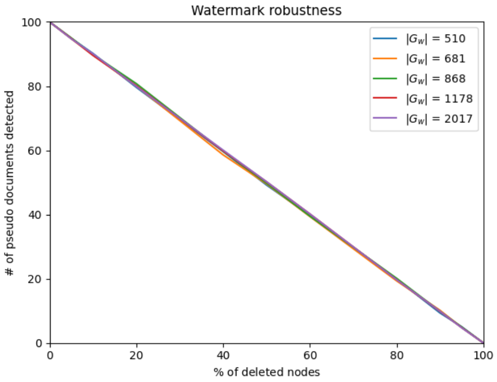

Figure 7: Deletion attack results.

Figure 7 shows that the watermark can withstand around 40% of documents being deleted before becoming undetectable. This result is for the range between 0 and 1000 watermarked documents. As shown, 20% of the watermarked pseudo nodes/documents are detected even if more than 80% of the nodes are deleted.

6 CONCLUSION AND FUTURE WORK

In this work, we proposed a new watermarking technique for graph databases inspired by Zhuang et al. [28]. Different from their technique, our solution does not suffer from heavy optimizations. We overcome their limitation due to heavy optimization by using a randomized approach during watermarking embedding. Due to randomization, the attacker cannot identify the pseudo document inserted into the database during embedding. We also implemented our technique to evaluate its performance using a real-world database. We analyzed that our technique is secure and robust against a range of attacks.

In our ongoing work, we are analyzing the robustness of our technique against more advanced attack scenarios such as insertion, update, and modification attacks. We are also working on automating the manual analysis for the watermarking process and parameter selection. We plan to compare our technique to other graph database techniques in terms of performance, usability, and robustness. We are also interested in applying our technique to other relational databases since many of them can be represented as a graph.

ACKNOWLEDGMENTS

Our work was supported by the European Union's HORIZON project DataBri-X (101070069).

REFERENCES

[1] 2023. The professional client, IDE and GUI for MongoDB. https://studio3t.com/

[2] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security*, William Enck and Adrienne Porter Felt (Eds.). 1615–1631. https://www.usenix.org/conference/usenixsecurity18/presentation/adi

[3] Rakesh Agrawal and Jerry Kiernan. 2002. Watermarking Relational Databases. In *International Conference on Very Large Data Bases, VLDB*. 155–166. https://doi.org/10.1016/B978-155860869-6/50022-6

[4] Ashima Anand and Amit Kumar Singh. 2021. Watermarking techniques for medical data authentication: a survey. *Multim. Tools Appl.* 80, 20 (2021), 30165–30197. https://doi.org/10.1007/s11042-020-08801-0

[5] Md. Asikuzzaman and Mark R. Pickering. 2018. An Overview of Digital Video Watermarking. *IEEE Trans. Circuits Syst. Video Technol.* 28, 9 (2018), 2131–2153. https://doi.org/10.1109/TCSVT.2017.2712162

[6] Erman Ayday, Emre Yilmaz, and Arif Yilmaz. 2019. Robust Optimization-Based Watermarking Scheme for Sequential Data. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2019, Chaoyang District, Beijing, China, September 23-25, 2019*. USENIX Association, 323–336.

[7] Malgorzata Bach and Aleksandra Werner. 2014. Standardization of NoSQL Database Languages. In *Beyond Databases, Architectures, and Structures (Communications in Computer and Information Science, Vol. 424)*, Stanislaw Kozielski, Dariusz Mrozek, Pawel Kasprowski, Bozena Malysiak-Mrozek, and Daniel Kostrzewa (Eds.). 50–60. https://doi.org/10.1007/978-3-319-06932-6_6

[8] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. 2012. Efficient Padding Oracle Attacks on Cryptographic Hardware. In *Advances in Cryptology - CRYPTO*, Vol. 7417. 608–625. https://doi.org/10.1007/978-3-642-32009-5_36

[9] Christian S. Collberg and Clark D. Thomborson. 2002. Watermarking, Tamper-Proofing, and Obfuscation-Tools for Software Protection. *IEEE Trans. Software Eng.* 28, 8 (2002), 735–746. https://doi.org/10.1109/TSE.2002.1027797

[10] Saad M. Darwish, Hosam A. Selim, and Mohamed M. El-Sherbiny. 2018. Distortion Free Database Watermarking System Based on Intelligent Mechanism for Content Integrity and Ownership Control. *J. Comput.* 13, 9 (2018), 1053–1066. https://doi.org/10.17706/jcp.13.9.1053-1066

[11] CFE Dr. Shannon Block. 2019. Why Amazon, Google, Netflix and Facebook switched to nosql? https://www.linkedin.com/pulse/why-amazon-google-netflix-facebook-switched-nosql-shannon-block-cfe

[12] Khaled Emam, Lucy Mosquera, Richard Hoptroff, and an O'Reilly Media Company. Safari. 2020. Practical Synthetic Data Generation. https://www.safaribooksonline.com/complete/auth0oauth2/&state=/library/view//9781492072737/?ar

[13] Tsvetomir Hristov. [n. d.]. Graph Database Watermarker(BEP). https://github.com/Elkozel/graph-database-watermaring-BEP

[14] Muhammad Kamran and Muddassar Farooq. 2018. A comprehensive survey of watermarking relational databases research. *arXiv preprint arXiv:1801.08271* (2018).

[15] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multim. Tools Appl.* 80, 5 (2021), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

[16] Jonathan Katz and Yehuda Lindell. 2014. *Introduction to Modern Cryptography, Second Edition.* CRC Press. https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269

[17] Stefan. Katzenbeisser and Fabien A. P. Petitcolas. 2000. *Information hiding techniques for steganography and digital watermarking.* Artech House, Boston.

[18] A Kerckhoffs. 1883. A. Kerckhoffs, la cryptographie militaire, Journal des Sciences Militaires IX, 38 (1883). In *Journal des sciences militaires.*

[19] Vidhi Khanduja. 2016. Dynamic watermark injection in NoSQL databases. *Journal of Computer Science Applications and Information Technology. Symbiosis* (2016), 1–5.

[20] Yingjiu Li and Robert Huijie Deng. 2006. Publicly verifiable ownership protection for relational databases. In *ACM Symposium on Information, Computer and Communications Security, ASIACCS*. ACM, 78–89. https://doi.org/10.1145/1128817.1128832

[21] William Lyon. [n. d.]. UK Companies Data. https://neo4j.com/graphgists/35a813ba-ea10-4165-9065-84f8802cbae8/

[22] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux journal* 239 (2014), 2.

[23] Arezou Soltani Panah, Ron G. van Schyndel, Timos K. Sellis, and Elisa Bertino. 2016. On the Properties of Non-Media Digital Watermarking: A Review of State of the Art Techniques. *IEEE Access* 4 (2016), 2670–2704. https://doi.org/10.1109/ACCESS.2016.2570812

[24] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. 2008. Watermarking Relational Databases Using Optimization-Based Techniques. *IEEE Trans. Knowl. Data Eng.* 20, 1 (2008), 116–129. https://doi.org/10.1109/TKDE.2007.190668

[25] Ta Minh Thanh, Nguyen Huu Thuy, and Ngoc-Tu Huynh. 2018. Key-value based data hiding method for NoSQL database. In *International Conference on Knowledge and Systems Engineering, KSE*. IEEE, 193–197. https://doi.org/10.1109/KSE.2018.8573334

[26] Ramarathnam Venkatesan, Vijay V. Vazirani, and Saurabh Sinha. 2001. A Graph Theoretic Approach to Software Watermarking. In *Information Hiding, International Workshop, IHW*, Vol. 2137. Springer, 157–168. https://doi.org/10.1007/3-540-45496-9_12

[27] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. 2020. Model Watermarking for Image Processing Networks. In *Conference on Artificial Intelligence, AAAI*. 12805–12812. https://ojs.aaai.org/index.php/AAAI/article/view/6976

[28] Xiaodan Zhuang, Xi Luo, and Letian He. 2022. Document Database Watermark Algorithm Based on Connected Graph. In *2022 International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT)*. 212–218. https://doi.org/10.1109/CCPQT56151.2022.00044

[29] Devriş İşler, Elisa Cabana, and Nikolaos Laoutaris. 2022. FreqyWM: Frequency Watermarking for the New Data Economy. *https://dspace.networks.imdea.org/handle/20.500.12761/1626* (2022).