ISSN: 0067-2904

A Review of Software Watermarking

Sarah H. Mnkash*, Matheel E. Abdulmunem
Department of Computer Science, University of Technology, Baghdad, Iraq

Received: 8/12/2019 Accepted: 12/5/2020

Abstract

The Internet is the hallmark of the age of technology and this technology is complemented by the presence of software which is one of the basic components of the operation of the network and it is used in almost all daily life aspects such as industry, commerce and others. Because the digital documents and objects can be easily replicated and distributed at an economically low cost and as the software is a type of digital object, the problem of software watermarking risen as related to how to protect data from piracy. Therefore, various techniques have been developed to protect codes from misusing and unauthorized alteration. Each of them is known as watermarking technology that protects data by inserting secret information into software, as an indicator of copyright ownership for this software. In this paper, the watermarked software will be explained in detail by defining the classification of watermarks software, models of attack, and methods used in software watermarking carried out by several researchers. It seems clearly that the method of ROP algorithm has given the highest accuracy results up to 100%

Keywords: Software Watermarking, Attack of software watermarking, ROP algorithm

مسح للعلامة المائية للبرمجيات

ساره حسين منكاش*, مثيل عمادالدين عبدالمنعم

قسم علوم حاسوب, الكلية العلوم, الجامعة التكنولوجية, بغداد, العراق

الخلاصة

الإنترنت هو عصر التكنولوجيا وهذه التكنولوجيا تكملها وجود البرمجيات التي تعد أحد المكونات الأساسية لتشغيل الشبكة ، والتي تستخدم في جميع جوانب الحياة اليومية تقريبًا مثل الصناعة والتجارة وغيرها. نظرًا لأنه يمكن نسخ المستندات والكائنات الرقمية بسهولة وتوزيعها بتكلفة منخفضة اقتصاديًا ، وبما أن البرنامج هو نوع من الكائنات الرقمية ، فإن مشكلة وضع العلامات المائية على البرامج حول كيفية حماية البيانات من القرصنة ، وبالتالي تم تطوير تقنيات مختلفة لحماية التعليمات البرمجية من إساءة الاستخدام والتغيير غير المصرح به. كل واحد منهم عبارة عن تقنية للعلامات المائية تحمي البيانات عن طريق إدراج هذه المعلومات السرية في البرنامج كمؤشر لملكية حقوق الطبع والنشر لهذا البرنامج. في هذه الورقة ، سيتم شرح برنامج العلامة المائية بالتفصيل من خلال تحديد تصنيف برنامج العلامات المائية ، ونماذج الهجوم ، والأساليب المستخدمة في العلامة المائية للبرنامج التي تم

*Email: cs.19.58@grad.uotecnology.edu.iq

تخصيصها من قبل الباحثين وأظهرت استنتاجاتهم أن هناك طريقة خوارزمية ROP أعطت عالية نتائج دقة تصل إلى 100٪.

1. Introduction

For many software companies and information technology industries, piracy of software has become a major issue, which was associated with the rapid evolution of these industries. Losses form piracy of software are rising each year, reaching $51 billion in the economic value [1]. The developers of software are interested in protecting their products' intellectual property from software piracy, i.e. preventing their code from being illegally reused [1]. A solution to these problems appeared in software watermarking as a technique used for integrating confidential information into the software text [2]. Such data may decide software ownership, so if the unauthorized utilization of the technology occurs, software copyright holders may have proof of infringement by deleting such confidential messages from an unauthorized copy [3]. Software's watermarking is a method of embedding a signature, meaning that it represents the owner reliably in the cover software. This allows developers of software to verify their copyright through extracting their signature from illegal copies. Researchers have developed a number of watermarking techniques over the past two decades that can be categorized into three major classes according to the extraction method, namely abstract, static and dynamic watermarking [4]. Most current watermarking methods attack a feature of the program that can assume many connections but, however, only hiding the watermark in one connection. Consider, for example, the technique of watermarks presented by an earlier work [5]. This technique changes the allocation of the record; although there are many customizations that fit the data of the program, only one is set to be the signature and is therefore utilized in the software selected. This method is similar to that suggested in another article [6], where the characteristic flipping of the base blocks is chosen among the many possible blocks. Both of the above-mentioned techniques are static due to only affecting program planning. Note that a static watermarking program only displays the watermark and excludes all other programs; this may help attackers rather than blocking them, not to mention the ease of planning the scheme while maintaining the functionality. Dynamic watermarking technologies exploit the links assumed by programs at the runtime, allowing many candidate organizations to coexist in the same program. Path-based technology, for instance, addresses the actions of branching programs at the runtime; the code executes various routes on various inputs, but only the particular inputs provide the route defining the signature outline [7]. Similarly, the bonding method gives multiple threaded software so that various correlations arise on how to solve the thread race conditions; it provides special inputs [8].

2. Literature Review

In this paragraph, we will review the range of research that has addressed Survey Software Watermarking.

A new string was included in individual parts of the program so that the dynamic behaviour of the string is distinct when given the correct input, and the watermark is encoded. This strategy is effective for an attacker against static analysis who has the capacity to implement the software, but it can very easily fail. Moreover, it is not appropriate for software wherever speed is critical [9].

Database watermark's goal creates powerful and continuous database watermarks. An image-based method is suggested as a watermark and this watermark is included in the database in two different attributes, one in the numeric attribute of the value and the other in the attribute time field in seconds. This approach can be applied to the numerical and categorical databases [10]

A new design of dynamic software is water-based on Return Oriented Programming (ROP). Software watermarks have design formats with watermark icon in well-structured data arrangements that look like normal data but it can be executed even the data almost similar normal data. Once it runs, the execution of the previously created ROP will retrieve the hidden watermark message. The ROP-based watermark technology is stealthier and flexible compared to technologies that have existed, since the watermark symbol is dynamically allocated in attacked data area, therefore the code is analysed based on these attacks. Ratings showed that this design not only achieves satisfactory performance and flexibility but also significantly reduces overhead watermark software [11].

An attacker's ability to identify a signature under abstract interpretation can be modelled as a property of completeness. It is considered that the attackers are abstract translators who can accurately monitor the characteristics they complete. Hiding a signature in the code matches its inclusion in terms of a semantic feature that can only be retrieved by attackers who complete it. In fact, no incomplete translator for a property that specifies the signature can detect, tamper with, or remove it. To provide a formal framework for modelling, at the semantic level, for software watermark methods and feature quality [12].

A novel dynamic watermarking method, Xmark, which is known unsolved mathematical problem referred to as the Collatz conjecture. This method has worked by transforming the selected conditional constructs (related to the software to be watermarked) with a control flow obfuscation technique based 0n Collatz conjecture. These obfuscation routines were built in a particular way such that they were able to express a watermark in the form of iteratively executed branching activities occurred during computing the aforementioned conjecture. Exploiting the one-to-one correspondence between natural numbers and their orbits computed by the conjecture (also known as the "Hailstone sequences"), Xmark's watermark-related activities were designed to be insignificant without the pre-defined secret input. It was shown that this method could remain robust even if a watermarked software was compromised via re-obfuscation using approaches like control flow flattening [13].

3. Software Watermarks

Software watermarking is unique among the techniques that can shield software from piracy, since it is not intended to avoid the piracy of software, but instead, it seeks to present evidence of an incidence of piracy [14]. A common aim of multimedia watermarking is to combat media piracy, for example copyright protection of movies in DVD format [14]. In computer science, it is indeed a common subject for research. Watermarking programs remain a relatively recent field. Although the objectives of watermarking multimedia and watermarking technology are similar in that they inject some additional information into items that are digitally encoded, watermarking methodologies are included in the program in order to preserve the operational implications of the program.

In most watermarking media, the watermark is not embedded in a simple microlayer. Rather, the only location where a watermark can be obscured is in the presence of a multimedia artefact [14]. In the program watermark, such modified watermarks are possible and are called Easter eggs [14]. Figure 1 shows software watermarking for embedding and detection algorithm.

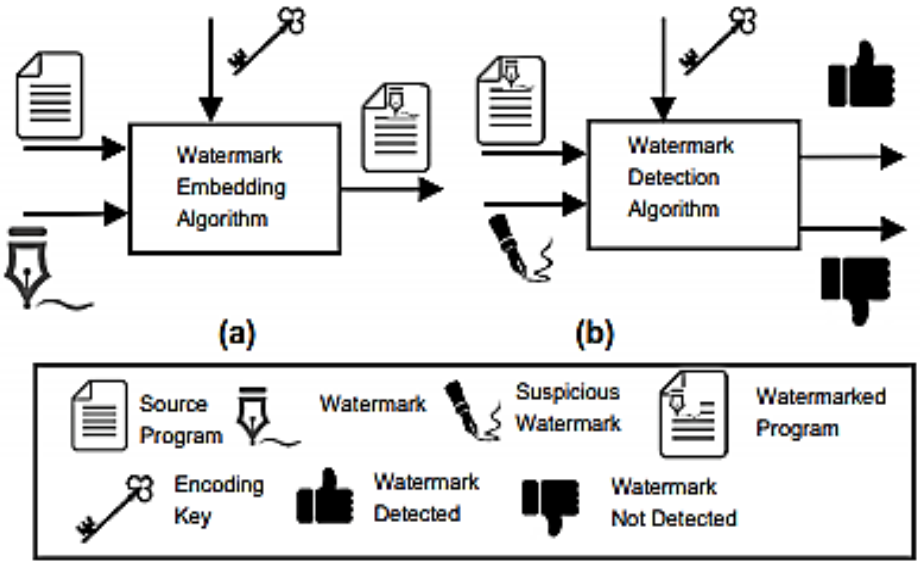

Figure 1: Software watermarking for embedding and detection algorithm. (a) watermark embedding algorithm (b) watermark detection algorithm [15].

3.1 Classification of Software Watermarks

There are different ways to classify software watermarks according to their tasks and characteristics. Below are several schemes of classification in the published literature. Software watermarks are categorized according to their functional objectives [16] as prevention signs, confirmation marks, permission marks, and confirmation marks. Blocking prevents the use of unauthorized marks of the software. Confirmation marks make a public statement to the program. Permission marks allow limited change or copying of the driver. Confirmation marks ensure that there is an end-user to authenticate the program. The watermarks of software can also be marked as static or dynamic by their extraction techniques [16]. The static watermark code is put in the symbols data area or text. Extracting this watermark does not require running the program. A dynamic software watermarking is inserted if the program object is executed. More precisely, in the dynamic watermark of the program,

However, certain symbols may be included rather than the watermark itself [16], because the watermark is expressed or extracted in case of running the program. Both robust and fragile software watermarking types exist. Robust watermarking may be extracted for programs even if they have been interpreted as hostile, accidental or semantic. To prevent unauthorized uses, watermarks have been utilized in systems and also in systems that make software public claims. Fragile watermark software is always to be decimated when the software is changed. These watermarks are used to verify the integrity of programs and systems, allowing for limited modification and copying [16]. Based on the features that a user can try; the software watermarking can be classified as invisible or visible for the software. If a watermark is included for a visual program, the watermarked program will generate some readable images, such as those of the logo and others. An invisible watermark is a software that can not appear to the end-user as a legible image but some algorithm can extract it, not being directly controlled by end-users. Based on whether the original software or watermarking is a watermark extractor input, a watermark software can be classified as either informed or blind [17].

A.Static and Dynamic Software Watermarking

The software watermarks are divided into two categories depending on the extraction technique [17].

1. ***A static software watermarking*** is included into the data area or text symbols. Extracting these watermarks do not require running the program. There are generally two kinds for static watermarks [17]; code and data watermarks. A data watermark is entered directly into the program's data area, while a watermark with an icon is entered in the program's code area. A simple code watermark flipping involves arranging some instructions in the program. Figure (2) illustrates a simple static watermarking system [17].

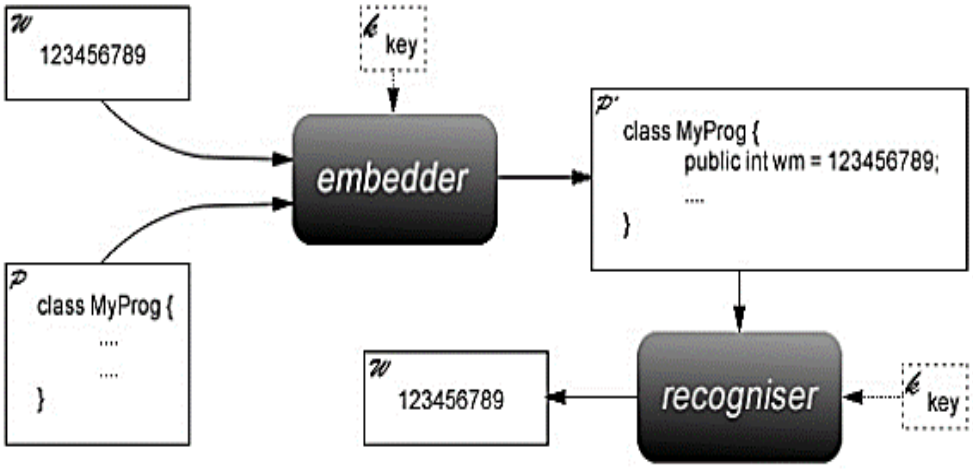

Figure 2-A simple static watermarking system. [18].

2. ***A dynamic software watermark*** is included in state to execute the software object. More specifically, in the dynamic watermark of the program, what is included is not the watermark

itself but some symbols. When running the program, they trigger the watermark to be expressed or extracted, as in the watermark of the dynamic data structure. There are three types of dynamic software watermark; white Easter watermarks, implementation dynamic watermarks, and the watermark of the dynamic data structure [19]. Figure-3 shows a system of simple dynamic watermarking.

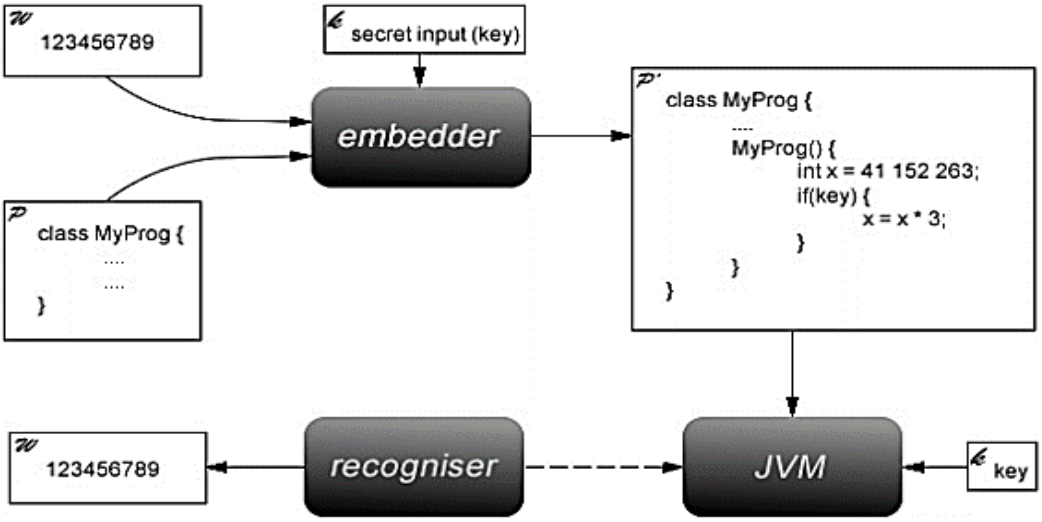

Figure 3-System of simple dynamic watermarking [20].

B. Robust and Fragile Software Watermarks

1. *Robust software watermarks.* It can also be obtained if they are interpreted as hostile, undesirable or indicative. Watermarks of this type have been utilized in preventive systems of untrusted or unauthorized uses for prevention, as well as systems that claiming ownership of the publicity of the program (confirmation) [21].

2. *Fragile software watermarks.* When the software has been changed, it will always be destroyed. Such watermarks are used in software integrity verification (affirmations) and in systems allowing limited modification and copying (permission) [21].

C.Visible and Invisible Software Watermarks

Software watermarks can be classified as visible and invisible [22], depending on characteristics that a software user can experience.

1.*Visible Software Watermark*. Any special input may trigger code to create a legible image such as a logo,.. etc., such watermarks are visible in the program (confirmation) or believing validity (affirmation) users [22].

2.*Invisible Software Watermark.* It appears to the end-user, not as a legible image, but can be extracted through some method that are not directly controlled by end-users. These are utilized for licenses and preventive measures [22].

3.2 Attacks on Software Watermarks

There are two ways to attack programs, namely malicious client attacks or malicious host attacks. Overall, the watermark of the program is designed to protect programs against malicious host attacks. There are four primary ways in the software to attack the watermark, defined as added attacks, subtraction attacks, malformed attacks, and attacks on recognition. In additional attacks, adversaries include a new watermark in the watermarked software, so that the original technology copyright owners cannot check the original watermarks ownership [23].

Opponents delete the watermark of the watermarked system by repelling attacks without disrupting the watermarked program's usability. The opponents ' goal in skewed attacks is to change the watermarks to stop copyright owners from reproducing them while preserving the software's usability [23].

1.*AdditiveasAttacks*: In the watermark, a new watermark code is added, so that the software's original copyright holders cannot assert their rights by adding their original watermark into the software [23].

2.*Subtractiveas Attacks*: Watermark code is disabled without influencing watermarked software's usability. Distortive attacks change the watermark to stop the copyright owners from removing it and still preserve the software's usability [23].
3.*Recognitionas Attacks*: The watermark detector or its inputs are updated or disabled to give a misleading result. An opponent, for instance, may say that "his" watermark detector is the one that should be used in a court check to prove ownership [23].

3.3 Software Watermarking Algorithms

The main watermarking algorithms currently available in the program will be listed in this paragraph.
1.*Basic block reordering algorithm*: It is the first algorithm for software watermark. The idea of this algorithm was to rearrange its basic blocks [24].
2.Register Allocation Algorithm: This method inserts a watermark in the program's interference graph [25].
3.Opaque predicate algorithms: The watermark and the opaque predicate are inserted in the dummy process [26].
4.Threading Algorithm: This algorithm depends on the inherent complexity of a multi-threaded program thread during the run [27].
5.Abstract interpretation algorithm: The watermark is embedded in values allocated to local integer variables during the execution of the program. These values can be determined by analysing the program within an abstract interpretation framework that allows the detection of the watermark even if only a part of the watermarked program is present [28].
6.Dynamic path algorithm: A watermark is placed during program operation by inserting a watermark in the branch structure. It depends on the fact that the branch structure is a primary part of the software and that it is difficult to analyse such a structure entirely because it has many program implications [29].
7.Graph-based algorithms: It is the first software watermarking based on graph application and it is known as the algorithm of VVS. It is a static watermarking algorithm for software. The first method of the dynamic graph is the CT algorithm. It embeds the watermark in graph data structure which is built during the execution of the program so it is a dynamic software watermarking algorithm [30].
8.Spread-spectrum algorithms: It is a vector program which changes each component of the vector by adding a small random amount [31].

4. Summary of Software Watermarking

In this paragraph, the methods used are briefly illustrated and the most comprehensive contributions of watermark software are shown in the Table (1) below.

Table 1-Summary of Software Watermarking Findings from Literature

Referewnce	***Algorithm***	***Year of Publicatio n***	***Outcomes***
Christian and Thomborson [32]	Dynamic graphic watermark	1999	The classification of software watermark techniques depends on how the tags are merged, retrieved, and attacked. In addition, it has introduced the watermark conceptual model for programs. However, the most interesting result is the new watermarking techniques family for software that includes tags within the dynamic data structures of the topology heap.
Palsberg et al [33]	Embed a watermark in dynamic data	2000	The experimental result shows that the watermark may be effectively achieved with modest increases in the size of code, execution times and use of heap space while keeping the watermark code robust

	structures		for different conversion attacks. To obtain a specific representation of watermarks, the retrieval time of the watermark is one minute per megabyte of heap space. The application is not designed to resist all possible attacks; to do this, it should be combined with other protection methods, such as jamming and tampering.
Agrawal and Kiernan [34]	Relational databases for Watermark	2002	This algorithm assumes that numeric attributes can withstand modifications of some less significant bits. Therefore, Tuples were first chosen to include the watermark. Then, certain parts of some of the selected group attributes were modified to include watermarks.
Collberg et al [6]	Encode the watermark in forwarding branches.	2004	There are two ways to encode watermark bits, as a string, if-statement and as a loop. Debugging code is used to ensure that only a subset of watermark pieces is necessary to restore the tag.
Zhao-Hong, and Jian-jun [35]	Software watermarking algorithm based on chaos	2006	The algorithm incorporates the methodology of anti-reverse engineering, the chaotic process and the concept of watermarks in the Easter Egg code. Analysis indicates that the algorithm resists various semiconductor transformation and has reasonable reverse engineering.
Nagra and Clark [9]	Threading software watermarks	2007	A new string is included in individual parts of the program so that the dynamic behaviour of the string is distinct when given the correct input, and it encodes the watermark. This technique is safe against static analysis, but for an attacker who has the capacity to implement the software, it can very easy to fail. Moreover, it is not appropriate for software wherever speed is critical.
Kamel and Albuwi [36]	Rtree data structure	2009	A system for watermarking Rtree data structure and its variants utilized by program execution. A thorough safety assessment and performance evaluation demonstrated the robustness of the embedded watermarks.
Brijesh and Udai [10]	Multi_ place watermarking in database security	2014	Database watermark's goal is creating powerful and continuous database watermarks. An image-based method is suggested as a watermark which contains two different attributes of one in the database, one in the meaning numeric attribute and the other in the time field of the attribute in seconds. The quantitative and categorical structures can be applied .
		2015	A novel design of dynamic software with

Ma et al [11]	Return Oriented Programming (ROP)		water-based return Oriented Programming (ROP). Formats of the watermark icon were designed in well-structured data arrangements that look like normal data but it can be executed even the data almost similar normal data. Once it runs, execution of the previously created ROP will retrieve the hidden watermark message. The ROP-based watermark technology is stealthier and flexible compared to technologies that have existed, since the watermark symbol the data area was dynamically allocated and therefore stopping attacks based on code analysis. Ratings showed that the design not only achieves satisfactory results and flexibility but also significantly reduces overhead Watermark software
Laftah and Jalil[37]	Concealment makes sense based on the value of the dihedral angle	2015	Logical concealment according to the double layer edge value which is also the distance between the two planes. The concealment began in the process of deciding. The core of the model is similar to the spider house constructing in terms of the average starting point, then starting to create the network to move in the clock direction and loop forming to follow the largest loop forming by using the same method of hiding data and creating more. The approach showed very good results with respect to an image update since there was no update to the original image and the error.
Dalla Preda and Pasqua [12]	A semantics-based approach of Software watermarking	2017	An attacker's ability to identify a signature under abstract interpretation can be modelled as a property of completeness. Attackers are considered abstract translators who can accurately monitor the characteristics they complete. Hiding a signature in the code matches its inclusion in terms of a semantic feature that can only be retrieved by attackers who complete it. In fact, no incomplete translator for a property that specifies the signature can detect, tamper with or remove it. The aim of this work was to provide a formal framework for modelling, at the semantic level, for software watermark techniques and quality features [12].
Ma et al	Collatz	2019	A new dynamic watermark scheme named Xmark, which depends on the collatz-conjecture utilizing noise control. This technique utilizes a sequence of Hailstone

[13]	Conjecture		to encrypt binary messages generated by Collatz guess. Though a heavily coded watermark identifiable through the storage of various Collatz walking routines through the same Hailstone sequence.
Ali [38]	3D watermark model based on the closest distance	2019	A 3D watermark model based on the closest distance. The watermarking path shifts much further than possible between several heads in the field before the data for the inclusion is complete. Based on latency and reliability, the proposed algorithm has achieved strong results. Visibility was calculated in the range of Square root error (RMSE) and the distance from Hausdroff (HD) which had good results. The proposed approach demonstrated resistance to engineering attack (translation, scaling and rotation) as well as appropriate resistance to signal processing attacks such as noise inclusion and rationalization.

5. Results and Dissections

In this research, many previous works that have used many methods of watermarking software were discussed. All of these methods have given good results, but it was identified a very reliable method, namely the return of directed programming resulted in 100% efficiency. A novel design of dynamic software was based on Return Oriented Programming (ROP). Once it runs, it executes the previously created ROP which will retrieve the hidden watermark message. The ROP-based watermark technology is stealthier and more flexible compared to technologies that have existed, since the watermark symbol in the data area is dynamically allocated and then away from attacks based on code analysis.

6.Conclusions

As a result of the tremendous development in the field of information technology and the internet, there has been a wide exposure to a flood of violations to circumvent and steal organized and unorganized information. Piracy of software is a global issue and has becoming increasingly important to software developers and suppliers. Software watermarking is one of many techniques for protecting copyright. From previous studies, it was found that the work proposed in 2015 gave high accuracy of up to 100% because it has used the ROP algorithm and achieved the exact code distribution. Through the given analysis, it seems that this paper has provided different views of software watermarking.

References

1. Collberg, C. and Thomborson, C. **2002**. " Watermarking, tamper-proofing, and obduscation-tools for software protection". IEEE Trans. Software Eng., pages 735–746.
2. Collberg, C., Carter, E., Debray, S., Huntwork, A., Kececioglu, J., Linn, C. and M. Step p. **2003**. "Dynamic path-based software watermarking", ACM SIGPLAN Notices, Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation, Vol. 39, Iss. 6, June.
3. Collberg, C., Jha, S., Tomko, D. and Wang, H. **2004**. UWStego: A General Architecture for Software Watermarking, Technical Report), available on http://www.cs.wisc.edu/ hbwang/watermark/TR.ps on Nov. 20.
4. Chen, X.; Fang, D.Y.; Shen,J.; Chen, F. Wang, W. and He. L. **2009**. "A dynamic graph watermark scheme of tamper resistance". In Proceedings of the 2009 Fifth International Conference on Information Assurance and Security - Volume 01, pages 3–6. IEEE Computer Society. ISBN 978-07695-3744-3.

5. Collberg, C.S. and Thomborson, C. **2000**. "Watermarking, TamperProofing, and Obfuscation-Tools for Software Protection. 2003," Technical Report 2000-03, Universityof Arizona.

6. Collberg, C.; Carter, E.; Debray, S.; Kececioglu, A.; Huntwork, C.L. and Step. M. **2004.** "Dynamic path-based software watermarking." Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation.

7. Patrick C. And Radhia C. **2004**. "An Abstract Interpretation based framework for Software Watermarking," POPL' 04, Venice, Italy. ACM.

8. Gaurav G. and Josef P. **2007**. "Software watermarking Resilient to Debugging Attacks," Journal of multimedia, vol. 2, no. 2. Academy publisher.

9. Nagra, J. and Clark T. **2007**. "Threading software watermarks." International Workshop on Information Hiding. Springer, Berlin, Heidelberg.

10. Brijesh, M. B. and Udai, P. R. **2014**. "A Novel approach as Multi-place Watermarking for Security in Database." arXiv preprint arXiv:1402.7341.

11. Ma, H.; Lu, K.; Ma, X.; Zhang, H.; Jia, C. and Gao, D. **2015**. "Software watermarking using return-oriented programming." Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. ACM.

12. Dalla Preda, M. and Pasqua, M. **2017**. "Software watermarking: a semantics-based approach." Electronic Notes in Theoretical Computer Science 331, pp 71-85.

13. Ma, H.; Jia, C.; Li, S.; Zheng, W. and Wu, D. **2019**. "Xmark: Dynamic Software Watermarking using Collatz Conjecture." IEEE Transactions on Information Forensics and Security.

14. Hachez, G. **2003**. "A comparative study of software protection tools suited for ecommerce with contributions to software watermarking and smart cards," Ph.D.dissertation, Universite Catholique de Louvain, Mar.

15. Dey, Ayan, Sukriti Bhattacharya, and Nabendu Chaki. **2019**. "Software watermarking: Progress and challenges." *INAE Letters*, **4**(1): 65-75.

16. Nagra, D.J., Thomborson, C. and Collberg, C. **2002**. "A functional taxonomy for software watermarking", In Proc. 25th Australasian Computer Science Conference 2002, ed.MJ Oudshoorn, ACS, pp. 177-186.

17. He, Y. **2002**. "Tamperproofing a software watermark by encoding constants," Master's thesis, University of Auckland, Mar

18. HAMILTON, James; DANICIC, Sebastian. An evaluation of the resilience of static java bytecode watermarks against distortive attacks. *IAENG International Journal of Computer Science*, 2011, **38**(1): 1-15.

19. Collberg, C. and Thomborson, C. **2004**. "Software watermarking: Models and dynamic embeddings," in Proceedings of Symposium on Principles of Programming Lan guages, POPL'99, pp. 311–324.

20. James Hamilton, **2010**. "Types of Software Watermark," Software Watermarking, July 2010. https://jameshamilton.eu/research/types-software-watermark

21. Nagra, J., Thomborson, C. and Collberg, C. **2002**. "Software watermarking: Protective terminology," in Proceedings of the ACSC 2002.

22. Narayanan, A. and Shmatikov, V. **2005**. "Obfuscated databases and group privacy,"in CCS'05, Alexandria, Virginia, USA, November 7–11, pp. 264–173.

23. Naumovich, G. and Memon, N. **2003**. "Preventing piracy, reverse engineering, and tampering," *Computer*, **36**(7): 64–71.

24. Nicherson, J., Chow, S. and Johnson, H. **2001**. "Tamper resistant software: extending trust into a hostile environment," in Proceedings of ACM Multimedia '01. ACMPress.

25. Van Oorschot, P. **2003**. "Revisiting software protection," in ISC 2003, ser. LNCS, vol.2851, pp. 1–13.

26. Pal, S. and Mitra, P. **2004**. "Case generation using rough sets with fuzzy representation," *IEEE Trans. On Knowledge and Data Engineering*, **16**(3): 292–300.

27. Palsberg, J., Krishnaswamy, J.S., Minseok, K., Ma, D., Shao, Q. and Zhang, Y. **2000**. "Experience with software watermarking," in Proceedings of the 16th Annual Computer Security Applications Conference, ACSAC '00. IEEE, pp. 308–316.

28. Pawlak, Z. **2005**. "Some remarks on conflict analysis," *European Journal of Operational Research*, **166**: 649–654.
29. Collberg, C., Myles, G. and Huntwork, A. **2003**. "Sandmark–a tool for software protection research," *IEEE Security and Privacy*, **1**(4): 40–49.
30. Collberg, C., Kobourov, S., Carter, E. and Thomborson, C. **2003**. "Errorcorrecting graphs for software watermarking," in 29th Workshop on GraphTheoretic Concepts in Computer Science, July.
31. Collberg, C. and Thomborson, C. **2002**. "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *IEEE Transactions on Software Engineering*, **28**: 735–746, Aug.
32. Collberg, Christian, and Clark Thomborson. **1999**. "Software watermarking: Models and dynamic embeddings." Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages.
33. Palsberg, J.; Krishnaswamy, S. ; Kwon, M. ; Ma, D. ; Shao, Q. and Zhang, Y. **2000.** "Experience with software watermarking." Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00). IEEE.
34. Agrawal, R and Kiernan, J. **2002**. "Watermarking relational databases." VLDB'02: Proceedings of the 28th International Conference on Very Large Databases. Morgan Kaufmann, 2002.
35. Zhao-Hong, L. and Jian-jun, H. **2006**. "DCT-domain fragile watermarking algorithm based on Logistic maps." Acta Electronica Sinica 34.12: 2134.
36. Kamel, I. and Albuwi, Q. **2009**. "A robust software watermarking for copyright protection," Computer&Security, vol. 28, no. 6, pp. 395- 409, Sep.
37. Laftah, M. M., & Jalil, L. F. (2015). Watermarking in 3D Model Using Dihedral Angle. *Iraqi Journal of Science*, *56*(4C), 3546-3553.
38. Ali, N.A. **2019**" Watermarking in 3D Models Using Depth Path" Iraqi Journal of Science, 2019, Vol.60, No.11, pp: 2490-2496