ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(21), DOI: 10.17485/ijst/2015/v8i21/79312, September 2015* ISSN (Online) : 0974-5645

A Performance Comparison on Characteristics of Static and Dynamic Software Watermarking Methods

Hyun-Il Lim[*]

Department of Computer Engineering, Kyungnam University, Changwon, South Korea; hilim@kyungnam.ac.kr

Abstract

With the recent advancement in computing environments, software plays increasingly important role in real world applications. The time and effort for developing software are also increasing. Software license agreements are required from users to protect such software as intellectual properties of developers. However, the cases of software theft or license infringement are increasing every year. To cope with this problem, there have been several researches for protecting software licenses. In this paper, we introduce several approaches for software watermarking methods, which are technical approaches for identifying software copyright information. We compare their characteristics and performances in several evaluation metrics. Software watermarking methods are expected to be practical approaches for protecting and detecting the cases of software theft in several applications.

Keywords: Software Copyright Protection, Software License, Software Watermark

1. Introduction

With the pervasive high-speed internet and various application programs in wide areas, it is essential to use computer systems in real-world environments. In addition, the role of software in computing environments became more important. In these environments, it requires much more time and efforts to develop new software. To protect such software as intellectual properties of its developers, software license agreements are required to users of software.

Software license is a mean to protect the rights for the properties for the software. However, the cases of infringing the rule of software licenses are increasing every year. From the reason, the economic loss caused by such infringements also increasing every year[1].

In this paper, we introduce several methods for software watermarking to identifysoftware copyright information and compare the performances of such software watermarks according to performance criteria.

This paper is organized as follows: In section 2, we introduce the concepts of software watermarking methods and the approaches of static and dynamic watermarks. In section 3, we explain the performance metric for evaluating software watermarking methods. In section 4, we compare the performances of static and dynamic software watermarking methods and conclude in section 5.

2. Software Watermark

2.1 The Concept of Software Watermark

Recently, there have been several technical approaches for protecting software licenses. Generally, watermarking techniques are used to embed copyright information in binary data represented as forms of digital media. This approach is frequently used for multimedia products, such as images, audios and videos. The method inserts copyright information into such multimedia products and recognizes the inserted information when it requires the information. If the copyright information of such products is uncertain, the inserted copyright can be an evidence to verify ownership of the products by recognizing the previously inserted information.

** Author for correspondence*

Software watermark[2-4,8] is an approach which is applied such techniques to the area of software. The insertion of copyright information to software can be an evidence of the ownership of the software. Software watermark is implemented by the following two functions:

embed (P, w. k) → P'

recognize (P', k) →w

The embed() function generates watermark embedded program from an initial Program P' with a watermark *w* and a key *k*. The recognize() function recognizes the inserted watermark *w* from the watermark embedded program with the previously defined key *k*. By applying these two functions, the copyright of software can be identified by recognizing previously inserted watermark from suspicious software.

Figure 1 shows the procedure for applying software watermarking to identify software copyright information. Before releasing an original program P, watermark w is embedded into the program by embed() function with a key. The watermark embedded program is generated and it is released to users. Afterwards, if it is suspicious that the software is used without permission in the other software, the suspicious software can be identified by recognizing the previously embedded watermark by using recognize() function with the secret key.

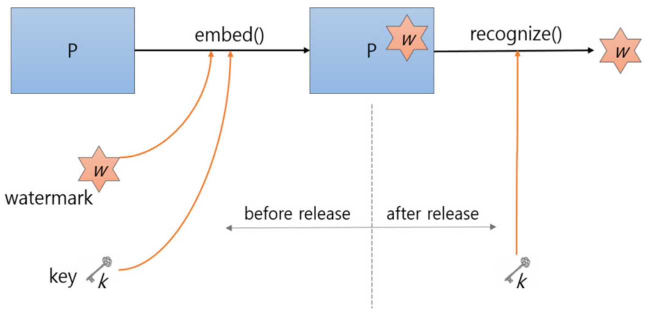

Figure 1. The procedure for applying software watermarking to identify software copyright information.

2.2 Static Software Watermark

Static software watermark is astatic approach to recognize previously embedded watermark from software by analyzing the software. This approach does not require executing programs to recognize watermarks. So, it is more convenient than dynamic approach because execution of software requires additional efforts to support various effects caused by individual execution environments.

2.2.1 Data Watermark

Data watermark[3] is the simplest way to embed watermark into software. This approach inserts data containing watermark information in specified location of software. For example, copyright information may be inserted in software code as follows:

String watermark = "Copyright © 2014. Watermark.";
The inserted copyright information can be used as a mean for identifying copyright owner of the software. This approach is very easy and simple to apply to software as it can embed and recognize watermark in a simple way. However, the embedded watermark can be revealed and easily removed by users or developers because the embedded watermark information is exposed as it is.

2.2.2 Code Watermark

Code watermark[5] is an approach to embed watermark information by relocating additional hidden information inside a program code. This approach can be applied in various ways according to methods of analyzing program structures. As an application of these approaches, the order of basic blocks of a program can be relocated by employing branch instructions to represent copyright information of the program through the order of basic blocks.

Davidson et al.[6] proposed an approach to embedding watermark by relocating the order of basic blocks. The watermark information is inserted by reordering the basic blocks in lexicographic order. The watermark information can be recognized by interpreting the relocation of the reordered basic blocks. The order of basic blocks are remained in its original order, the previously embedded watermark can be recognized. This type of watermark is not exposed outside of a program. Therefore, this approach is stealthier than the previously described data watermark. However, this approach requires additional program code for inserting watermark information. It also requires analyzing the orders of basic blocks in a program to recognize embedded watermark.

2.3 Dynamic Software Watermark

Unlike static software watermark, dynamic software watermark is a method for recognizing embedded watermarks by analyzing the states or the behaviors of a program during the program execution. This approach requires additional analysis for modeling the behaviors and execution states of programs to specify and recognize watermark in program execution. So, the method inserts watermarks into states or behaviors of an analyzed program. Then, it can recognize the inserted watermark by interpreting the states or the behaviors of the program during the program execution.

2.3.1 Easter Egg Watermark

Easter egg means hidden messages or functions, which are invoked when predefined special input is entered in software. For example, especially in game software, Easter egg is included as a hidden stage entered when a secret key input is entered. Easter egg watermark[3,4] is a method of inserting code for expressing software watermark of copyright information, which is invoked when predefined specific input is entered. The watermark information may contain copyright information or specific image representing the ownership of software. So, the recognition of Easter egg embedded in software can prove the ownership of the software.

In this approach, the embedded watermark will not appear if previously defined secret input is not entered. So, it may be difficult to find embedded watermark during program execution. However, the watermark itself is openly exposed in its program code. Therefore, the watermark may be easily found by analyzing program code.

2.3.2 Data Structure Watermark

Data structure watermark[5] is a method of inserting watermark information into runtime states expressed by data structures constructed during the program execution. This approach organizes previously designed data structures for representing copyright information when the program encounters a specified runtime state or input. For example, this approach may make variables or linked lists containing copyright information when a user continuously clicks specific area of screen during program execution. Such data structures can encode watermark information to represent copyright information of software. So, if the generated data structures are interpreted successfully, the ownership of the software can be proven.

Data structure watermark is very stealthy and resilient, because the watermark cannot be found by users if the defined input sequence or the definition of data structures is not exposed. However, this approach requires additional structuresto embed program code for generating data structure of watermark. It also requires additional analysis to interpret the generated data structures to identify the ownership of software.

3. Performance Criteria of Software Watermark

To apply software watermark techniques in real world applications, the approach should have practical performances in embedding and recognizing watermarks in programs. There are several evaluation criteria for measuring the performance of software watermark. In this paper, we applied the evaluation criteria presented in Myles et al.[7] to compare the performance characteristics of software watermarks presented in the previous section. In this section, we describe the 6 evaluation criteria for evaluating performance of software watermarks.

3.1 Credibility

In the situation when software copyright information has to be confirmed, it is desirable that the embedded watermark should be recognized easily. In addition, the recognized watermark should be credible enough to confirm the copyright information without false positives, which can decrease the reliability of the watermark's results. Therefore, the recognized watermark should be an obvious evidence for confirming software copyright information.

3.2 Data-rate

To identify the copyright information in the future, software watermark should be embedded to software before releasing the software. Embedding of watermark into software requires additional spaces in the software. If the additional space is too large as compared to the size of original software, the embedded watermark will be easily noticed or altered. Therefore, the ratio of additional space for embedded watermark as compared to the original program should be low enough to hide the existence of software watermark.

3.3 Stealth

The embedded watermark should be able to be recognized by the owner of software to verify software copyright information. However, if the watermark is easily disclosed or removed by other users or developers during program execution or modification, the method cannot be practically applied in real-world applications. Therefore, watermark should not be noticeable to the other users with high stealth.

3.4 Part Protection

To practically apply software watermark in real world environments, the embedded watermark should be remained with its copyright information after user's arbitrary modification. To meet this property, it is more effective to embed software watermark distributed over the whole program areas as compared to embed in specific local areas of a program.

3.5 Overhead

If the time and effort for embedding and recognizing software watermark is too high, the watermark method cannot be used in practical application. In addition, if insertion of software watermark degrades execution performance of software, the watermark cannot be used in real-world applications. Therefore, software watermark should be efficient in embedding and recognizing software watermark. In addition, software watermark should have less influence to execution performance of software.

3.6 Resiliency

When software is used by users or newly developed by developers, it may be modified or changed in various ways such as optimization or obfuscation. In such environments, embedded watermark should be remained without loss of copyright information. Therefore, the resilience of watermark is essential for practical use of software watermark to protect software ownership when software is altered by various modifications.

4. Performance Comparison of Software Watermarks

In this section, we compare the performance of characteristics of software watermarks with the evaluation criteria presented in the previous section according to performance characteristics of each approach. The performances of individual watermarks may change according to execution environments or implementation method of each watermark. So, we try to compare the performances of software watermarks with the general characteristics of software watermarks. Table 1 summarizes the performance comparison results of software watermarks with the 6 evaluation criteria for software watermarks.

4.1 Comparison of Static Approaches

4.1.1 Credibility

In data watermark, watermark is directly embedded into the code of software itself without hiding the existence of watermark. So, it can be an obvious evidence of the copyright of software. However, the embedded watermark can be easily recognized by analyser, because embedded watermark is opened as it is. In code watermark, watermark is embedded inside of program code. When a watermark-embedded program is used and modified by users, its embedded watermark may also be affected by user's modification. In such cases, code watermark may not be recognized successfully.

Table 1. Performance comparison results of characteristics of software watermarks.

Measurement Criteria	Data Water-mark	Code Water-mark	Easter egg Watermark	Data Structure Watermark
Credibility	High	Medium	High	High
Data-rate	High	Medium	Medium	Low
Stealth	Low	High	Medium	High
Part Protection	Low	High	Low	Medium
Overhead	Low	Medium	Medium	High
Resiliency	Low	Medium	Medium	High

4.1.2 Data-rate

In data watermark, the size of embedded watermark is very small because there is almost no additional space for data except for its watermark itself. In code watermark, however, it is required to modify the code of program and insert additional code to embed watermark.

4.1.3 Stealth

In data watermark, the embedded watermark can be easily discovered by analysing program code because the embedded information is exposed as its original form. In code watermark, however, the embedded watermark is hidden into inner part of software code, so it is required to analyse the program code to discover the existence of watermark. Therefore, the code watermark is stealthier than the data watermark.

4.1.4 Part Protection

In data watermark, it is hard to distribute watermark over the whole area of software to protect the copyright of a whole program because a watermark is embedded in a specific local area of software code. In code watermark, however, watermark may be distributed over several areas of software code in relocating information of program code to embed the watermark.

4.1.5 Overhead

In data watermark, the procedures for both inserting and recognizing watermark hardly affect the execution performance of software, because the embedding of data watermark does not require additional operations in executing software. In code watermark, however, it is required to perform additional operations to embed or recognize watermark. In addition, the embedding of watermark may affect execution performance of software. So, code watermark costs a little overhead for the procedures.

4.1.6 Resiliency

In data watermark, embedded watermark may be directly affected by user's modifications or obfuscations because the watermark is openly exposed without hiding the existence of watermark. In code watermark, watermark is encoded into hidden area of software and it is not exposed to users or developers. So, code watermark is more resilient than data watermark.

4.2 Comparison of Dynamic Approaches

4.2.1 Credibility

Both of Easter egg watermark and data structure watermark embed watermark intothe execution states of program and the embedded watermark is executed when a predefined secret key is entered. So, the recognition of embedded watermark is obvious evidence of software copyright without false positives.

4.2.2 Data-rate

To apply dynamic software watermark, it is required to embed additional code to describe the behaviors of runtime execution of software and execute watermark when control flow reaches predefined program state. Especially, data structure watermark requires additional complex code to generate specific data structures for representing watermark information. So, data structure watermark require additional space for embedding and recognizing watermark.

4.2.3 Stealth

Dynamic watermarksare stealthier than static approaches because they embed watermarks into program's execution states of programs. In Easter egg watermark, embedded watermark can be easily discovered by code analysis because the watermark is openly exposed in program code. In data structure watermark, however, watermark can hardly be discovered because the watermark is generated only when the program reaches a specific runtime state during the program execution.

4.2.4 Part Protection

In Easter egg watermark, watermark is generally embedded into a specific area of program code without distributingthe watermark over whole area of program. So, the watermark is embedded as a hidden code, which is only invoked when a predefined specificinput is entered. On the other hand, in data structure watermark, watermark is embedding in dynamically generated data structures. So, the code for generating watermarks may be distributed to several areas of software to improve the part protection property of watermark.

4.2.5 Overhead

In data structure watermark, it costs much time and efforts to design data structure and implement code for generating the data structures for representing watermark. It also takes additional runtime overhead for executing software to generate additionally embedded data structures. In Easter egg watermark, however, watermark may be easily embedded to appear only if previously defined secret key is entered. So, it does not affect execution time overhead for generating watermark.

4.2.6 Resiliency

In data structure watermark, watermark is embedded as a form of data structures generated only if the control flow of a program reaches specific runtime states of a program. Because the watermark is not exposed to users or developers if the control flow does not reach the specific states of a program, the watermark is hardly discovered by users. So, the watermark is resilient to program modifications or obfuscations. In Easter egg watermark, however, watermark is exposed openly to users and it can be easily discovered by users through code analysis. So, watermark can be directly affected by user's modifications.

5. Conclusion

As the importance of software increases, the software licenses as intellectual property have become more important. There have been various researches to deal with the increasing cases of infringements of software licenses. In this paper, we introduced the concepts of software watermarking methods, which can confirm the software copyright information by embedding watermark before releasing software. We also presented several approaches of static and dynamic software watermarking methods, which have various characteristics in software code and runtime execution.

In this paper, we introduced the performance evaluation metric for software watermark. Then, we compared the performance of the characteristics of each software watermark. Generally, static software watermark are easier to use than dynamic approaches. It is because they do not require program execution and they embed watermark directly into software code. So, the watermark may be easily discovered and removed by code analysis. However, dynamic software watermarks are hardly discovered by users or developers because watermarks are contained in runtime program states during program execution. So, it is required to embed additional code for generating runtime watermark. Moreover, dynamic watermarking methods take additional overhead for maintaining embedded watermarks.

In applying software watermark methods in real-world applications, it is important to optimize the methods according to individual execution environments and software characteristics. In the future, it is expected that software watermarking approaches will be applied in various environments for protecting software and detecting cases of infringements of software licenses.

6. Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0024658).

7. References

1. Business Software Alliance. 2011 BSA Global Software Piracy Study. 9th ed. 2012.
2. Monden A, Iida H, Matsumoto K, Inoue K, Torii K. A practical method for watermarking java programs. Proceedings of the 24th Computer Software and Applications Conference; Taipei; 2000. p. 191–7.
3. Collberg C, Thomborson C. Software watermarking: Models and dynamic embeddings. Principles of Programming Languages (POPL'99); 1999. p. 311–24.
4. Collberg C, Thomborson C. Watermarking, tamper-proofing and obfuscation - tools for software protection. IEEE Trans Software Engineering. 2002 Aug; 28(8):735–46.
5. Collberg C, Thomborson C. On the limits of software watermarking. Department of Computer Science: University of Auckland; 1998. Technical Report No: 164.
6. Davidson RL, Myhrvold N. Method and system for generating and auditing a signature for a computer program. US Patent 5,664,191. 1996.
7. Myles G, Collberg C. The evaluation of two software watermarking algorithms. Software Practice and Experience. 2005 Aug; 35(10):923–38.
8. Hamilton J, Danicic S. An evaluation of static java bytecode watermarking. Proceedings of the World Congress on Engineering and Computer Science; 2010 Oct.