EMBEDDING A SOFTWARE WATERMARK USING POISSON DISTRIBUTION

D. Seetha Mahalaxmi¹, S. Viswanadha Raju and A. Vinay Babu³,

¹Associate Professor, Dept. of CSE, JNTUHCEH E-mail: lakshmi.dsml@gmail.com
²Professor and HOD of CSE Jagityal, JNTUH E-mail: sravu.jntu@gmail.com
³Principal, JNTUHCEH E-mail: avb122@gmail.com

ABSTRACT
Now days, the information transfer from one media to other media became very common. The information transfer can be the data, software, or programs, and so on. In this scenario, a lot of attacks are possible. In this paper various software watermarking algorithms were discussed and finally embedding of a watermark using Poissons Distribution was given.

Keywords: Software watermarking, Embedding, Obfuscation, Tamper-proof, Recognition, Poisson Distribution.

1. INTRODUCTION
Software watermarking is embedding a watermark in the software which authenticates the owner of the software. A lot of research work was carried in embedding of a watermark in software. In this paper various software algorithms that were developed were discussed. In the end of the section the new method of embedding software watermarking using Poisson Distribution was given.

2. OVERVIEW
In order to protect the software from attacks, or from revealing the secret information, various techniques were proposed, out of which some of them are: software watermarking, tamper proofing, obfuscation. Software watermarking means to embed secret information in software so that the intellectual property right of the software is protected. Tamper proofing, where it becomes difficult for an attacker to alter the given software, and obfuscation, it is difficult for an attacker to reverse engineer the given software.

It is important to prevent the attacker from attacking software, because perfect copies of software will be developed by an attacker and uses it for their own purpose.

3. IMPORTANCE OF SOFTWARE WATERMARKING
Software watermarking embeds secret information into cover message. The need for software watermarking can be well understood by the following scenario: There are three participants in this scenario, Alice, Bob, and Charles. Alice is the author of software who wants to sell her entire program or main logic of the program to her customer Charles. Bob is a pirate who attempts to illegally copy and resell or reuse the Alice's program. If Bob pirated the program, Alice has to prove that her software has been stolen or misused.

In order to protect from pirates the Alice is going to embed a watermark function, and even to increase the complexity of finding the watermark, the Alice has produced a function q which is an obfuscated function or a cropping function. An obfuscation function produces code which is semantically equivalent to the original, but harder to understand.

Watermark embedded must be resilient to various attacks, and must be able to give approximate information even in the case of attacks, and be proving that Bob is involved in Piracy act.

4. STUDY OF SOFTWARE WATERMARK ALGORITHM
A lot of research work was carried out in this area. In this section, a brief discussion on some of the techniques was given.

4.1. Basic Block Reordering Algorithm
The software watermark algorithm [9], embeds a watermark into a program by reordering the basic blocks of the program. A basic block is a set of sequential instructions with a single entry point and single exit point. The reordering maintains the original flow of execution so that the function of the original program is unchanged. Checking the order of this group of blocks enables us to extract the inserted watermark.

Attacks are easily possible on this algorithm, which makes the original watermark to be easily destroyed. The algorithm can be enhanced using the opaque predicate to establish false dependencies among basic blocks, making it difficult to remove them.
4.2. Register Allocation Algorithms
Qu and Potonjak [18][19], discusses on the watermark techniques to constraint problems such as graph-colouring problems. It aims to watermark solutions to the graph-colouring problems to protect their intellectual properties. The graph-colouring problem, concerns allocating as fewer colours as possible to the varieties of a graph so that no vertices connected by an edge in the graph receive the same colour.

In [14], the QP algorithm was implemented by register allocation. The QPS algorithm [14], which was proposed allows robust extraction of watermark in the absence of attacks. But Qp and QPS algorithms are not suitable for software watermarking of architecture-neutral codes in the presence of determined attackers.

4.3. Spread-Spectrum Algorithms
Speed-spectrum watermarking [7], represents the data of a document as a vector and modifies each component of a vector with a small random amount. This small amount is called watermark. Such watermarks can be recognized by correlation with the extracted watermark signal. The spectrum algorithm discusses on extraction, watermark insertion, and watermark testing.

In SHKQ algorithm [22], which makes use of spread-spectrum algorithm, code is viewed not as a set of sequential instructions, but as a statistically object. Sahoo and Collberg [20], have implemented the SHKQ algorithm in the software watermark research tool SandMark. In this paper the method of overloading to increase the frequency of patterns in case where code insertion and code substitution were discussed.

In [4], a software watermarking method using graph depth as a signal was proposed. In this algorithm, a vector from running a program is extracted. The call graph depth is measured at distinct points during the execution of the program to be watermarked on certain particular input. At end, the program code is modified so that its call graph depth is changed, such that it expresses the watermark when this input is given to the program.

4.4. Opaque Predicate Algorithms
An opaque predicate is a predicate whose outcome is known to the developer who uses this predicate and the code. Its value is difficult to analyze. Using this concept, in [12][13], a method to insert watermark into a dummy method never intended for execution was introduced. This dummy method is guarded by an opaque predicate.

In [10], class file concept was used, i.e. it tries to make the relations between class files unclear by distributing the methods between class files and destroying abstractions.

[1], a method for introducing watermark into opaque predicates and an opaque predicate into its associated dummy method, was given. [15], the above algorithm was implemented and tested using SandMark were given.

4.5. The Threading Algorithm
A threaded watermarking algorithm was proposed in [17], and implemented it for Java byte code. This algorithm takes the advantage of the intrinsic randomness for a thread to run in a multithreaded program. This algorithm claims resilience. The process of embedding watermark is as follows: firstly it creates multiple threads of execution. Secondly locks are added to ensure that only small subset of the possible paths are actually executed by the watermarked program. The watermark is embedded in those execution paths.

4.6. The Abstract Interpretation Algorithm
Abstract interpretation algorithm [2], was used to embed a watermark to values assigned to designated local variables during program execution. These values can be determined by analyzing the program under an abstract interpretation framework, enabling the watermark to be detected even if only part of the watermarked program is present.

4.7. The Metamorphic Algorithm
Metamorphic transformations Thaker [23], were used to increase the diversity of the software and embed a fingerprint into the software. The pattern for the watermark to be inserted into software is such that, to attack the watermarks fairly easy.

4.8. Dynamic Path Algorithm
A Dynamic-path algorithm was proposed [3], was used to insert a watermark in the runtime path-based branch structure of a program to be watermarked. This algorithm inserts the watermark into the suitable points, which can be referred to be the branches, by observing their execution. In order to trace the watermark, check the branch instructions to extract the watermark.

4.9. The Mobile Agent Watermarking
Mobile Agent watermarking [10], uses software watermarking in mobile agents. It aims to guarantee the integrity of the execution of the mobile agents. Watermark is embedded into a mobile agent, and then it is transferred to the agent's result during its execution. When the agent returns to the origin host, the results of all the hosts involved are verified in order to assure the integrity of the mobile agent.

4.10. Graph-Based Algorithm
Graph-Based algorithm, watermark is encoded in a special graph G. There are two types of graph based
algorithms, static and a dynamic graph based. VVS algorithm [25], give static way of embedding the watermark, while CT algorithm [8][5] gives the dynamic way of embedding the watermark.

4.11. Birthmarks
Software birthmarks [24], are a unique set of characteristics of a Java class file for detecting software theft. It is based on the assumption that if a class file has the same birthmark as another class file, these two class files have the same source, i.e. one is copy to another. In Dynamic birthmark [16], the dynamic control flows of a program were given. But the only condition is, the birthmark should be resilient to semantic preserving transformations.

5. POISSON DISTRIBUTION
Poisson distribution is used as a means for embedding a watermark in a given software [21]. Poisson distribution assigns probabilities to the number of occurrences. Lambda (\( \lambda \)), is specified in Poisson distribution which helps to calculate mean number of occurrences.

The Poisson distribution is helpful in giving the probability of events, to model the number of events occurring within a given time interval. For example: the number of phone calls received by a telephone operator in a 10-minute period, and the number of flaws in a bolt of fabric.

The formula for the Poisson distribution is given by:

\[
p(x, \lambda) = \frac{e^{-\lambda} \lambda^x}{x!} \quad \text{for } x = 0, 1, 2, \ldots
\]

5.1. Implementation of
As \( \lambda \) value is fixed, a random variable is the value of customers identity number. When the customer enters the identity number, the probability value is calculated using Poisson distribution formula. This value is embedded by the owner in the software, at the receiving end this value is once again re-calculated by the receiver, and is compared with the secret code encoded in the watermarked program, if they are equal then only the actual program is going to be executed.

Embedding of a watermark by using Poisson distribution, is easier to implement, economically effective such as memory requirements and the cost of the implementation, and which is stronger to withstand the reverse engineering attacks.

There are already various tools that have developed for providing Tamper-resistance such as Software tamper resistance [1], Dynamic Path Based Watermark [1], and so on. When compared embedding of a watermark using Poisson Distribution makes it stronger from reverse engineering.

6. COMPARISON WITH BINOMIAL DISTRIBUTION
In Binomial distribution is similar to the Poisson Distribution, but in binomial distribution both the values that they are \( \lambda \) and the customer details were to be produced, but in Poisson Distribution customer identity is randomly generated, which is making the embedding of a software by using Poisson distribution more stronger than embedding the software with binomial distribution.

7. CONCLUSION
Software watermarking is one technique to protect the copyright of the owner. In this approach various techniques that have been developed in order to protect the software watermark were discussed. Watermarking is a technique where we are going to provide security for the coded program and the data it contains. The watermarked program is prone to various types of attacks such as additive attack, distortive attack, subtractive attack and reverse engineering. Here we have shown an application of mathematical formula Poisson distribution and embedded the value of the same in our watermarked program. By an implementation of such a technique, we can prove that the watermarked program is free from the reverse engineering.

REFERENCES


