SOFTWARE WATERMARKING USING FIXED SIZE ENCODING AND RANDOM DUMMY METHOD INSERTION

AZYAN YUSRA BINTI KAPI @ KAHBI

A dissertation submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia

MAY 2011

“Dedicated to my beloved family and friends, without their understanding, supports, and most of all love, the completion of this work would not have been possible.”

ACKNOWLEDGEMENT

First and foremost, I would like to thank Allah because of His blessings; I would be able to successfully complete this dissertation. My word of appreciation goes to Associate Professor Dr. Subariah Ibrahim for her priceless supervision, inspiring discussion and fruitful collaboration. I am thankful for all her invaluable hours to provide constructive critics, enthusiasm, immerse knowledge and continuous feedback. Without her continued support and patience, this dissertation would not have been the same as presented here.

I am also indebted to Universiti Teknologi MARA for funding my Master study. It is my sincere hope that the knowledge and experience gained here can be put to good use and further contribution in the future.

My thanks also extend to my friends, for their enlightening companionship and encouragement of trudging through all the moments from down to up the hill in the run to complete this Master program. I would not have done it without the help and motivation from all of you.

To my family, no words can describe my gratefulness for always being there despite of the distance. They showered me with love and compassion and enrich my life like no other. They are the source of comfort and kept me focus the priorities in life and therefore, this work is dedicated to them.

ABSTRACT

Recently, the rise of software piracy has become rampant and a major concern among software developers. The global software industry lost about USD 50 billion in 2008. One of the techniques that can be used to discourage piracy is watermarking, by embedding developer's watermark into software which can later be extracted to prove ownership. During the last few years, different algorithms were produced and developed to hide the watermark inside software. This study analyzes software watermarking algorithms that exist in the literature and then identified a dummy method algorithm is suitable for watermarking. In addition, this study enhances dummy method insertion technique in embedding and recognizing the watermark in Java class files. The enhancement includes fixed size encoding scheme and random dummy method insertion. The proposed fixed size encoding scheme used hash function that can produce a fixed size watermark bit sequences. Random dummy method insertion selects a dummy method at random from a collection of dummy methods. Finally, this study analyzes the enhancement of dummy method insertion technique using two different measures, namely data-rate and resilience of the watermarking algorithm. In term of data rate, the results show that encoded watermark for proposed encoding scheme is always fixed even though size of watermark character is increased. In term of resilience, experimental results show no similarity between class files and thus survived from collusion attack compared to previous method.

ABSTRAK

Kebelakangan ini, peningkatan cetak rompak perisian semakin berleluasa dan menjadi perhatian utama di kalangan pembangun perisian. Industri perisian global telah kehilangan sekitar USD 50 bilion pada tahun 2008. Salah satu teknik yang boleh digunakan untuk mencegah cetak rompak perisian ialah tera air, dengan membenamkan tera air pembangun perisian yang kemudiannya boleh diekstrak untuk membuktikan hak milik. Sejak beberapa tahun lalu, algoritma yang berbeza telah dihasilkan dan dibangunkan untuk menyembunyikan tera air di dalam perisian. Kajian ini menganalisis algoritma perisian tera air yang telah wujud dalam kesusasteraan dan mengenalpasti bahawa algoritma kaedah semu sesuai untuk perisian tera air. Selain itu, kajian ini meningkatkan teknik penyisipan kaedah semu dalam pembenaman dan pengenalpastian tera air dalam fail Java. Peningkatan yang dilakukan termasuk skim pengekodan bersaiz tetap dan penyisipan kaedah semu secara rawak. Skim pengekodan bersaiz tetap yang dicadangkan menggunakan fungsi cincang yang dapat menghasilkan turutan bit tera air yang tetap. Penyisipan rawak kaedah semu memilih kaedah semu secara rawak daripada koleksi kaedah semu. Akhirnya, kajian ini menganalisis peningkatan teknik penyisipan kaedah semu menggunakan dua ukuran yang berbeza, iaitu kadar data dan ketahanan algoritma pembenaman tera air. Keputusan menunjukkan kadar data tera air yang dihasilkan untuk skim pengekodan yang dicadangkan adalah sentiasa tetap walaupun saiz tera air meningkat. Dalam hal ketahanan, hasil eksperimen bagi beberapa fail Java yang telah dibenamkan tidak menunjukkan kesamaan dan oleh itu, selamat dari serangan kolusi berbanding teknik terdahulu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays, software has become a part of human's daily life to ease many tasks in the software industries, e-commerce and many more. As the usages of the software are growing rapidly, the rise of software piracy has become a major concern for software developers.

Among the software, Java applications that have been sold to users and distributed through the internet also suffered from piracy. While Java becomes a popular language in software industry and education, the advantages of Java which are platform-independent and its portability has create problem towards Java's user. Platform-independent means that once Java source code is written and compiled, it can be run anywhere in any platform.

The process of reverse-engineering is become easier, thus competitors can copy other person's work easily. This could bring benefits to competitors since it is time consuming and reduce cost without having to create the algorithm themselves. Furthermore, competitors and other people could claim other developer's program as their own products.

In order to make Java application executes in any platform, Java source code needs to be translated into Java class file. Based on Lim *et al.* (2009), Java class file contains Java byte code and understandable by Java Virtual Machine (JVM). An attack to this class file can easily be made by reverse-engineering or de-compilation of the class file itself. Thus, extracting the source code from the class file is possible by many tools that can be found in the internet.

As the technology become rampant, the process of copying other person's work become easier and it causes piracy to become crucial issues. According to the study done by the Business Software Alliance (BSA) in 2009 software piracy has caused USD 50 billion lost in the global software industry, whereas USD 368 million lost in Malaysia. In previous literature, there are many techniques to prevent software piracy. Code obfuscation, hardware based solution, checksums and watermarking are some of the example of the techniques that exist in the literature mentioned by Naumovich and Memon (2003). The details on software protection techniques were discussed in Section 2.2 of Chapter 2.

Software watermarking is one of the techniques that were used to prevent piracy by hiding the developer's information inside the software as stated by Cappaert *et al.* (2008). They stated that the information then can be retrieved and used in the future to prove ownership of the original developer.

1.2 Problem Background

Generally, in software industry and education, many people tend to copy algorithm in the software and to make it worse, they claim it as their works. Sivadasan and Lal (2007) stated that most of algorithms in software can easily be stolen by other competitors in the industry using many techniques and available tools. Zhu *et al.* (2005) also mentioned the fact that software piracy has become more and more important issue in today's business due to loss of million dollars in the industry.

Since many parties have suffered from software piracy, Pervez *et al.* (2008) indicated that many attempts are made to protect copyright in software. Software watermarking is one of the attempts where the copyright information is embedded in the software itself.

In software watermarking, developer embeds a unique identifier into software using specific tool that implements software watermarking algorithm. Watermarked software which contains the original software and developer's information will then be produced and distributed to the users whether for personal or commercial uses. When the developer noticed that someone is copying his software by claiming it as theirs, developer could come out with a proof. In this situation, the proof is the developer's information hidden inside the software. By entering specific input to the tool that was used originally in the embedding process, the developer could retrieve his information from the software. During the recognition process, the tool must have the ability to recognize the watermark's value in the software. The developer could declare that the software is originally developed by him and he is the one that embed the copyright information inside the software.

As there are many known techniques nowadays to protect software, more and more attacks have also become viable. This statement supported by Stern *et al.* (1999) and they claimed that it is impossible to secure the copy of digital document but it is possible to discourage the piracy.

In term of Java application, Monden *et al.* (2000) claimed that even though the Java application has a watermark embedded in it, it is easy to remove the watermark and embed a new watermark. This will replace the old watermark with the attacker's watermark information. Thus, the copyright of the original author is not permanently embedded in the application itself and cannot be used to prove ownership in future.

Many researchers found that even though watermark can prove ownership, Curran *et al.,* (2003) argued that most of watermark that can prove ownership usually cannot survive from various attacks. Although removal attack is the criteria that need to be given more concerned, some other attacks such as additive, subtractive, collusion and decompile-recompile attack are still vulnerable to the software. Further descriptions of these attacks were described in Section 2.5 of Chapter 2.

Up until now, many algorithms such as Qu-Potkonjak (QP) algorithm, opaque predicate algorithm and many others were introduced in the literature and some of the algorithms were described in Section 2.4 of Chapter 2. Despite all that, Myles *et al.* (2004) summarized that unfortunately most of the algorithms on software watermarking are not well-described, not being implemented and evaluated yet. Dummy method insertion technique by Monden *et al.* (2000) is one of the existing software watermarking algorithms that has several disadvantage. After several of experiments have been carried out, the disadvantages of dummy method have been discovered by Myles *et al.* (2004). In their study, they have tested two algorithms which are dummy method insertion algorithm and Davidson–Myhrvold

(DM) algorithm (Davidson and Myhrvold, 1996). Both of the algorithms have been evaluated according to the six different properties mentioned in their study.

In their evaluation, DM algorithm reveals a high credibility, satisfactory in data-rate and 50% survival rate towards resiliency. In case of dummy method algorithm, they have pointed several pros and cons in term of part protection, data-rate and also has 70% survival rate towards resiliency. The following paragraphs describe in more detail about these three terms.

Generally in the dummy method insertion technique, all the developer's information is hidden inside the dummy method. In term of part protection, the technique or algorithm must be able to fully protect the watermark so that the whole watermark is spread throughout the entire class file. Both of DM algorithm and dummy method insertion algorithm hide the watermark in a single method. Thus, Myles *et al.* (2004) concluded that DM algorithm and dummy method insertion algorithm can be considered as poor, since any alteration involved in the statement of the method will destroy the watermark.

High data rate represents a good point in the algorithm as it can hide a large portion of watermark within the class file. In term of the data rate, dummy method insertion algorithm does not have any difficulties compared to DM algorithm. This is because DM algorithm embeds the watermark in the largest control-flow graph (CFG) in application and depends on its size. As for dummy method algorithm, since the algorithm prepared the space for dummy method according to the size of the watermark, it has no difficulties in embedding large size of watermark. Thus, no matter how large the watermark's size, the dummy method could provide spaces for the watermark. But, in contrast, the larger the watermark's size, the longer dummy method's instruction will be produced. In this situation, instructions in the dummy method become longer than expected and hence will create suspicions from the attacker. Thus, the dummy method algorithm has a disadvantage in term of data rate,

since it has the ability to hide various sizes of watermark, but then it leads to the weakness of the dummy method.

Resilience can be defined as the capability of the technique in securing the watermark, resist and resilient to the attack imposed upon them. There are four types of attacks that have been highlighted by Myles *et al.* (2004) which are additive, subtractive, collusion and distortive attack. In addition to classify the attack, Gupta and Pieprzykalso (2007) have come out with seven different attacks towards the watermark in class file. However, Myles *et al.* (2004) highlight the disadvantage in term of resilience which is collusion attack towards dummy method. The collusion attack consists of performing different watermark in the same class file. After decompiling and comparing both of the watermarked class file, several instructions in the class file can be seen noticeably by the attacker. The only difference in the class file can be considered as the watermark. This caused dummy method to be discovered by the attacker and still need to be improved in term of collusion attack.

In this study, in formulating a research problem, a series of analysis has been done, but using a different test file from what Myles *et al.* (2004) have been used. After the analysis, the same problems were found in the three aspects mentioned by them which are part protection, data-rate and resilience. The discussion and analysis of this study can be referred in Section 2.6 of Chapter 2 and the results of the analysis were presented in Section 4.22 Chapter 4.

In this study, there are two criteria in the dummy method insertion technique that need to be improved. The criteria are in term of data rate and the collusion attack that are needed to be tested towards the watermarked class file.

1.3 Problem Statement

Java application has suffered from piracy and many algorithms have been introduced especially in the area of software watermarking to discourage piracy. One of the existing techniques which have been described in previous section is dummy method insertion algorithm. The dummy method insertion algorithm is then enhanced by Arboit (2002) by inserting opaque predicate to the calling method. Then, the credibility of dummy method technique also has been improved by Akbar (2010). However, the technique also has some flaws in other criteria that need to be improved. Problem with the dummy method insertion algorithm has been described in previous section.

In this study, after conducting an analysis and several backgrounds study in the literature, several questions have been formulated. Followed by this problem, the following questions have to be satisfied in conducting this study:

i. How to discover potential values of software watermarking algorithms that are going to be enhanced?
ii. How to make embedded watermark in the Java class file become not easily seen or less noticeable to the attacker?
iii. How to avoid increasing instructions in a dummy method during the embedding process?
iv. How to produce a fixed size of watermark bit sequences?
v. How to compare the proposed method with the previous technique in term of data-rate and resilience?

1.4 Dissertation Aim

This study is aim to discourage piracy by helping the Java developers to prove ownership towards their code on the future and enhance the dummy method insertion technique so that it become less noticeable to the attacker.

1.5 Objectives

In completing this study, there are three objectives that need to be achieved. The objectives are as follows:

i. To analyze existing technique in software watermarking.
ii. To design and implement an encoding scheme for watermark to be embedded in software.
iii. To enhance a watermarking technique that is less noticeable to attacker.

1.6 Dissertation Scope

The scope of the dissertation includes the following areas:

i. The dissertation is focus on placing watermark in Java class file.

ii. The performance of the technique is evaluated using a set of watermark properties which are credibility, data rate, stealth and resilience.
iii. Only five dummy methods were created in this study.
iv. Test files are taken from sample code provided in Java Development Kit (JDK) version 6.

1.7 Significance of the Dissertation

Generally, this study proposed a technique in embedding and recognizing a watermark in Java class file. The advantages and benefits in embedding and extracting the watermark in the application will contribute to original developer of the Java class file. Besides that, by placing the watermark in the program, the developer's information will remain embedded in the class file and other person cannot claim the program as their works. This is because if a person tries to copy the program, the original developer can prove his/her ownership by extracting the watermark from the program and prove that it is his/her. This scenario will help in reducing the number of software piracy.

In more details in term of significance, this study aims to enhance the technique in software watermarking which is dummy method insertion technique. By enhancing the algorithm according to both of data rate and resilience criteria, the dummy method can disguise as part of the methods in the class file, instead of its unused function. By all means, the dummy method or watermark that is embedded in Java class file will be less noticeable to attacker that performs decompile-recompile attack. In term of data-rate, by using the proposed method, the size of instructions in dummy method will be in a fixed number.

Thus, the proposed method can be used in proving copyright of the Java class file's developer. In this case, it is hope that the enhanced dummy method algorithm can be used in recognizing the original developer in future demand.

1.8 Organization of Report

This study consists of seven chapters. The chapters are organized according to different works that involved in this study. The detailed organization of this report is described in following paragraphs.

This section presents how this report is organize in different chapters. **Chapter 1** of this report consists of overview of the study, problem background, problem statement, objectives, scope and significance of this study.

Chapter 2 of this report presents a review of the literature related to the area of software watermarking. It discusses software protection technique in details that includes software watermarking, type of watermark and several attacks towards them, functionality, several problems in the previous literature regarding the technique and current solution in software watermarking.

Chapter 3 is consists of wide description on research methodology, which provides a rich discussion about the flow of this research. This includes how the operational and experimental work has been carried out for the study.

Chapter 4 is the discussion on analysis conducted in early phase of this study. This includes the results of experimental process and the comparisons of different technique in software watermarking. The results discussed in this chapter are used to formulate a problem in existing technique in software watermarking.

After that, **Chapter 5** discussed designs of proposed method in detail. Designs include two processes which are fixed encoding scheme and dummy method development.

Results on the proposed method will be compared with previous method in **Chapter 6**. A watermark property on credibility, data rate, stealth and resilience is tested on dataset defined in Chapter 3.

Chapter 7 is the conclusion of overall chapter and future works in the related area of software watermarking will be discussed in order to provide a better quality in future study. This includes recommendations for further study.

REFERENCES

Akbar, Z. (2010). Watermarking Java Programs using Dummy Methods with Dynamically Opaque Predicates, *Computing Research Repository (CoRR) 2010 informal publication.*

Arboit, G. (2002). A method for watermarking java programs via opaque predicates, *Procedding of the Fifth International Conference on Electronic Commerce Research (ICECR-5)*. October 23- 27, 2002. Montreal, Canada.

Business Software Alliance. *Sixth Annual BSA Global Software Piracy Study PC Software Piracy Rates and Losses 2004–2008*. United States, 2009.

Cappaert, J., Preneel, B., Anckaert, B., Madou, M., and Bosschere. K. D. (2008). Towards Tamper Resistant Code Encryption: Practice and Experience. *Proceedings of the 4th international conference on Information security practice and experience* (ISPEC'08), Liqun Chen, Yi Mu, and Willy Susilo (Eds.). Springer-Verlag, Berlin, Heidelberg, 86-100.

Curran, D., Hurley, N. J., and Cinnéide, M. Ó. (2003). Securing Java through Software Watermarking. *Proceedings of the 2nd international conference on Principles and practice of programming in Java, 2003. (PPPJ '03)*. Computer Science Press, Inc., New York, NY, USA, 145-148.

Collberg, C. (2003). *Sandmark algorithms.* University of Arizona, Department of Computer Science, Tech. Rep., Jul. 2003.

Collberg, C., Myles, G. and Huntwork, A. (2003). Sandmark—A Tool for Software Protection Research. *IEEE Security and Privacy 1,* 4 (July 2003), 40-49.

Collberg, C., Carter, E., Debray, S., Huntwork, A., Kececioglu, J., Linn, C. and Stepp. M. (2004). Dynamic Path-Based Software Watermarking. *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation (PLDI '04).* ACM, New York, NY, USA, 107-118.

Collberg, C., Huntwork, A. and Carter, E., (2004). Graph Theoretic Software Watermarks: Implementation, Analysis, and Attacks. *Proceeding of the 6th International Information Hiding Workshop, 2004*. May 23-25, 2004. Toronto, Canada.

Collberg, C. and Thomborson, C. (1999). Software Watermarking: Models and Dynamic Embedding. *Proceeding of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Language (POPL99)*. January 20-22, 1999. San Antonio Texas, USA. 311-324.

Collberg, C. and Thomborson, C. (2002). *Watermarking, Tamper-Proofing, and Obfuscation-Tools for Software Protection*. IEEE Transactions on Software Engineering, 735-746, August, 2002.

Cousot, P. and Cousot, R. (2004). An Abstract Interpretation-Based Framework for Software Watermarking. *In Conference Record of the 31st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Programming Languages, Venice, Italy, January 14-16, 2004*. ACM Press, New York, U.S.A. 2004.173-185.

Davidson, R. and Myhrvold, N. (1996). *Method and system for generating and auditing a signature for a computer program*, US Patent 5, 559, 884.1996.

Drape, S. (2009). Intellectual Property Protection using Obfuscation. *Research Project sponsored by Siemens AG, Munich.* 2009.

Fukushima, K. and Sakurai, K. (2004). A Software Fingerprinting Scheme for Java Using Class files Obfuscation. In: Fukushima, K. and Sakurai, K. *Information Security Applications Lecture Notes in Computer Science*. Berlin/Heidelberg Springer. 1991-2009; 2004.

Gupta, G. and Pieprzyk, J. (2007). Software Watermarking Resilient to Debugging Attacks (2007). *Journal of Multimedia*, 2, 10-16, Apr 2007.

Hamilton, J. (2008). *Static Source Code Analysis Tools and their Application to the Detection of Plagiarism in Java Programs.* Department of Computing at Goldsmiths University of London (2008).

Hamilton, J. and Danicic, S. (2010). An Evaluation of Static Java Bytecode Watermarking. *Proceedings of the World Congress on Engineering and Computer Science 2010* WCECS 2010, October 20-22, 2010, San Francisco, USA

Jamal, Z. and Wang, H. (2009). On the Analysis of Dynamic Software Watermarking. *Proceeding of 2nd International Conference on Software Technology and Engineering (ICSTE).* October 3- 5, 2010. San Juan, Puerto Rico. 26-30.

Lim, H., Park, H., Choi, S. and Han, T. (2009). A Method for Detecting the Theft of Java Programs through Analysis of the Control Flow Information. *Journal of Information and Software Technology* 51, September 9, 2009. 1338-1350.

Malik, S., H., K., Khan, A., Khalil, S. and Amjad, S., (2009). Evaluating Effectiveness of Tamper-proofing on Dynamic Graph Software Watermarks. *International Journal of Computer Science and Information Security IJCSIS* 2009. 6(3).

Memon, J., M., Khan, A., Baig, A., Shah, A. (2007). A Study of Software Protection Techniques. *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. 249-253.*

Monden, A., Iida H., Matsumoto K., Inoue K., and Torii K. (2000). A Practical Method for Watermarking Java Programs. *Proceeding of The 24th Computer Software and Applications Conference (compsac2000), Taipei, Taiwan.*

Myles, G., and Collberg, C. (2004). Software Watermarking Through Register Allocation: Implementation, Analysis, and Attacks. *Department of Computer Science University of Arizona, Tucson, AZ, 85721, USA.*

Myles, G., Collberg, C., Heidepriem, Z. and Navabi, A. (2005). *The Evaluation of Two Software Watermarking Algorithms.* Journal of Software – Practice and Experience. 35, 923–938

Myles, G., and Collberg, C. (2006). *Software Watermarking Via Opaque Predicates: Implementation, Analysis, and Attacks.* 6, 2 April, 2006. 155-171.

Nagra, J. (2006). Collusive Attacks against Software Watermarks. *Annual International Technical Conference of IEEE TENCON Region 10 Conference*. 14-17 November, 2006. Wan Chai, Hong Kong.

Nagra, J. and Thomborson, C. (2004). Threading Software Watermarks. *Proceedings of 6th International Workshop on Information Hiding, LNCS Volume 3200, Springer-Verlag.*

Nagra, J., Thomborson, C. and Collberg, C. (2002). A Functional Taxonomy for Software Watermarking. *Proceeding Twenty-Fifth Australasian Computer*

Science Conference (ACSC2002), Melbourne, Australia. CRPIT, 4. Oudshoorn, M. J., Ed. ACS. 177-186.

Naumovich, G. and Memon, N. (2003). *Preventing Piracy, Reverse Engineering, And Tampering*. Computer , 36(7), 64- 71, July 2003.

Palsberg, J., Krishnaswamy, S., Kwon, M., Ma, D., Shao, Q. and Zhang, Y. (2000). Experience with Software Watermarking. *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC '00)*. IEEE Computer Society, Washington, DC, USA, 308.

Pastuszak, J., Michalek, D. and Pieprzyk J. (2001). Copyright Protection of Object-Oriented Software. *Lecture Notes in Computer Science*, *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology table of contents.* 2288. 186 – 199,

Pervez, Z., Noor-ul-Qayyum, Mahmood, Y. and Ahmad, H.F. (2008). Semblance Based Disseminated Software Watermarking Algorithm. *23rd International Symposium on Computer and Information Sciences, 2008. ISCIS '08.* 27-29 October, 2008. 1-4.

Sivadasan, P. and Lal, S. P. (2007). JConstHide: A Framework for Java Source Code Constant Hiding. *Computing Research Repository CoRR*(*2009*) *informal publication.*

Stern, J. P., Hachez, G., Koeune, F. and Jacques, J. (1999). Robust Object Watermarking Application to code. *In Information Hiding*, Springer-Verlag, 1999.

Sun, G. and Sun X. (2010). Software Watermarking Based On Condensed Co-Change Graph Cluster. *Journal of Information Technology*. 9, 949-955.

Venkatesan, R., Vazirani, V. V. and Sinha, S., (2001). A Graph Theoretic Approach to Software Watermarking. *Proceedings of the 4th International Workshop on Information Hiding,* Springer-Verlag.

Zhao, H. (2002) Watermark Attacks [PowerPoint slides]. *Retrieved from ENEE739M Multimedia Comm. & Info. Security(S'02).*

Zhu, J. Q., Liu Y. H. and Ke, Y. (2009). A Robust Dynamic Watermarking Scheme Based on STBDW. *WRI World Congress on Computer Science and Information Engineering.* 7, 602-606.

Zhu, W. F. (2007). Concepts and Techniques in Software Watermarking and Obfuscation. *Thesis of Doctor of Philosophy in Computer Science*. The Department of Computer Sciences, University of Auckland, New Zealand.

Zhu, W., Thomborson, C. and Wang, F.Y. (2005). A Survey of Software Watermarking. *Conference of Intelligence and Security Informatics IEEE ISI-2005*, May 2005. Atlanta, Georgia. 454-458.

Zhu, W. and Thomborson, C. (2006). Algorithms to Watermark Software through Register Allocation. *Digital Rights Management: Technologies, Issues, Challenges and Systems, First International Conference DRMTICS 2005.* 31th October – 2nd November, 2005. Sydney, Australia. 180–191.