Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 2, February 2017, pg.156 – 169

Software Piracy Solutions

Laith Talib Rasheed

laithtalib77@yahoo.com

Physical Education & Sport College, Diyala University, IRAQ

Abstract

Most trade organizations face problems with software piracy. As a result there are many developed systems are available in markets to deal with this problem, but the fact that most of these systems do not offer an appropriate solution. To tackle software piracy a variety of solutions have been proposed. These solutions can be classified as either deterrent or preventive. Deterrent solutions reply to the fear of the consequences of getting caught. The solution is successful if an individual abstain from criminal behavior due to the perceived threat or fear of sanctions. Preventive solutions make use of current technology to increase the cost of the actual act of piracy. A deterrent solution relies on an individual fear of getting caught and does not directly increase the cost of the actual act of pirating. It is a mechanism put in place to discourage the act of piracy by imposing sanctions if the act is carried out and detected. In the United States which is the first country in the World deal with the deterrent solutions take form in several intellectual property rights laws.

1 Introduction

To tackle software piracy a variety of solutions have been proposed. These solutions can be classified as either deterrent or preventive. Deterrent solutions reply to the fear of the consequences of getting caught. The solution is successful if an individual abstain from criminal behavior due to the perceived threat or fear of sanctions. Preventive solutions make use of current technology to increase the cost of the actual act of piracy. These solutions can either be hardware based or software based and includes such technologies as tamperproof CPUs and software encryption [1]. Deterrent and preventive solutions will be explored further in the following sections. The issues associated with software piracy are not obvious to everyone, which could be due in part the non-exclusionary nature of a computer application [2]. To illustrate, suppose Alice has a copy of a popular video game on her computer. Alice can make a copy of the game and give it to Bob so he can play it on his computer. Now, both Alice and Bob own copies of the game which makes the computer game non exclusionary. On the other hand, Alice and Bob computers are exclusionary objects because only one of them can own each computer at a time. The exclusionary nature of the physical computer makes it clear who the property belongs to. This is not the case with intellectual property such as software. Even though the unethical nature of software piracy might not be obvious, the concerns are certainly not new. One of the major concerns with published literature, such as articles or books, is plagiarism. Within the software industry plagiarism is also a concern, but identifying and proving that a section of an application is stolen is far more difficult than with a published piece of literature. The difficulty in detecting software theft can mainly be attributed to the format in which software is distributed. For example, in the case of source code theft, the stolen code could be compiled using a different compiler which will yield an executable that looks different from the original. In addition, the economic impact for the company whose application was stolen can be severe. Software companies often make a significant portion of their revenue prior to the release of a competitor product. If a portion of their application is stolen the competitor is able to decrease production time and enter the market sooner. The second ethical issue is the illegal redistribution of the software. It is generally the case that pirated copies of software are distributed a significant discounted price, while still including all of the original functionality. Again, there can be an economic impact associated with this act. The ramifications associated with piracy propagate throughout the software industry. The obvious victims are the software companies themselves. However, the more peripheral victims are also not often recognized. Many pirates are undeterred by reports of financial losses suffered by software producers due to piracy. This could be because they do not see the trickle effect of the monetary losses. Many think of software producers as large companies which generate significant revenue, forgetting that the individuals who work for those companies feel the effects of piracy through decreased job opportunities or even lost jobs.

In 2001, up to 105,000 jobs, $5.3 billion in wages, and $1.4 billion in tax revenues were lost because of piracy in the United State alone [3].

2 Deterrent Solutions

A deterrent solution relies on an individual fear of getting caught and does not directly increase the cost of the actual act of pirating. It is a mechanism put in place to discourage the act of piracy by imposing sanctions if the act is carried out and detected. In the United States which is the first country in the World deal with the deterrent solutions take form in several intellectual property rights laws. The question of how these laws can be used to protect software has been debated for many years. The difficulty in devising the proper protection is rooted in categorizing software which can be a product, a service, or even mixture of both [4]. Currently, four intellectual property rights laws can be applied in the protection of software. These laws include copyright, patent, trademark and trade secret.

2.1 United States Copyright Law

Under the 1978 United States Copyright Act, as described by Tavani [5], a work must meet three requirements to receive protection. The requirements are originality, fixation, and expression. In general, any work which is original has a tangible form and a fixed in a medium can be protected under copyright. In other words, one cannot copy right an idea, but the tangible expression of the idea can be copy righted. Unfortunately, computer software is not fixed in a tangible medium like that of literacy works. This was the major difficulty which prior to 1980, made software ineligible for copy right protection. In 1980 the law was amended to address the nodes of computer software. The concept of literacy work was extended to include programs, computers and databases which exhibit authorship. The amendment defines a computer program as a set of statements or instructions to be used directly in a computer in order to bring about certain results. This addition has made it possible to copy right a program if it can be shown that the program contains an original expression of ideas and not simply the ideas. Additionally, the amendment ensures the protection of the source, object and executable code. There are two doctrines associated with the Copy right Act [6]. The first is fair use. Fair use permits the limited use of another person's copy righted work for the purpose of criticism, comment, news reporting, teaching, scholarships, and research. Use of the work outside of this violates the holder's rights. The second doctrine is first sale. The first sale doctrine allows the purchaser of a legally obtained piece of work to sell, rent, or give away the work without obtaining permission from the copy right holder. The first sale doctrine applied to software is not as straight forward as other works protected under the USA

Copyright Act. The main aspect which causes confusion, and has led to contradictory decisions in the courts, is that software under the End User License Agreement (EULA) is license, not sold. Many EULAs specifically state that resale is prohibited. Thus, the doctrine of the first sale does not apply to most software purchases.

2.2 Digital Millennium Copyright Act

The Digital Millennium Copyright Act (DMCA) was signed into law October 18, 1998. The DMCA made significant changes to the United States copyright law to address the changing needs associated with the digital age. One of the main focuses of the act was to address the treaties signed in December 1996 at the World Intellectual Property Organization (WIPO) Geneva Conference. The WIPO treatises required legal means against the act of circumventing anti-piracy measures and any technology which enabled the circumvention. Section 1101, circumvention of copyright protection systems, of the DMCA [7] outlines a few permitted exceptions to the anti-circumvention provision. These include nonprofit libraries, archives and academic institutions under special circumstances. Additionally, it is legal to reverse engineer protected software to conduct encryption research, assess product interoperability and to test computer security systems as long as it does not constitute copyright infringement and the person legally has the right to use a copy of the software. The anti-circumvention provisions of the DMCA have lead to a variety of unforeseen consequences which often hinder legitimate activities. For example, the DMCA has been used to stop the presentation and publication of research on security vulnerabilities in many products. Additionally, through the introduction of copy protected CDs, the DMCA can be used to prevent the fair use doctrine [8].

2.3 United States Patent Law

The United States Patent Law provides legal protection to individuals who create an invention or process [8]. A patent provides the inventors with exclusive rights to make, use, or sell the invention for 17 years. There are two basic requirements for an invention or discovery to the patentable: 1. it must be new and useful or a new and useful improvement. It must satisfy the following:

- The invention must have some usefulness or utility

- The invention must be novel

The invention must be not obvious to a ‘person of ordinary skill in the art’ who is familiar with the prior art. Due to the algorithmic nature of a software program, software was initially intelligible for patent protection. The first software patent was granted in 1981 for a program which assisted in converting rubber into tires. Currently about

10,000 new software patents are issued each year despite considerable debate over the appropriateness of patenting software [5].

2.4 United States Trademark Law

Kizza [8] describe a trademark to be a word, name, phrase, or symbol that identifies a product or service. They are often used by consumers to choose between competing products. Thus the United States Trademark Law helps ensure that the quality associated with a mark used by a business actually represents the quality expected by the consumer. The laws give the owner of a trademark the ability to prevent others from using the same or similar mark to promote their products. Under United States law there are three categories of trademarks which are protected in 10 year increments:

- Service mark: Used in the sale or advertising of a service.

- Certification mark: Used as a verifier or authentication of a product, service or group who offer a service?

- Collective mark: Used by a group of people to indicate membership. The only protection software actually receive through trademarks is if a pirate is deterred by the difficulty of passing off copies of well-known software.

2.5 United States Trade Secret Law

Unlike the three previous legal protections, United States Trade Secrets have no federal protection. All laws designed to protect trade secrets are at the state level and thus offer varying degrees of protection depending on the state. A trade secret consists of information used by a business or company which is of strategic importance in providing an actual or potential economic advantage over competitors. This information may be a formula, a design process, a device, or trade figures. Owners have exclusive rights to the secret but only for as long as the secret is maintained [8]. Applying trade secret law to software is difficult. Often what makes the software valuable is a particular technique or algorithm used. Because this information is released with the software, reverse engineering techniques can often be used to discover the secrets.

3 Preventive Solutions

Once the correct law is chosen there is the additional difficulty of enforcing the law. It is the responsibility of the intellectual property owner to ensure that their rights are not

violated. To this end, a variety of preventive solutions have been devised. These include audits of companies as well as hardware based and software based techniques.

3.1 Audits

Organizations such as the Business Software Alliance (BSA) perform audits to verify that corporations are not using illegal software. An audit involves taking an inventory of all material related to the software on the computer systems. Such material includes [6]:

- All media for installation
- All manuals and reference documentation
- All license documentation
- All documents proving the legitimacy of the software such as invoices

unfortunately, auditing does not necessarily identify an unknown software pirate or unethical programmer. It can help companies identify illegal practices by their employees or identify companies who are not purchasing the required number of licenses. However, the technique is ineffective at detecting the theft of algorithmic secrets. Additionally, the technique can identify if a company is guilty of piracy, but from the information the source of the piracy may not be obvious.

3.2 Hardware Based Techniques

Special purpose hardware is commonly used in proof of ownership, to provide secure data storage and to provide a secure execution context for security sensitive applications [10]. Such hardware is typically more cumbersome for the user and more expensive for the software vendor than software based techniques.

3.2.1 Dongles

A dongle is a hardware device distributed with software. Possession of the device proves ownership of the software. A dongle typically connects to an I/O port and computes the output of a secret function. While running, the software periodically queries the dongle. If the communication fails or the result of the query is incorrect, the software reacts appropriately [10]. There are two major drawbacks associated with the use of a dongle:

- Cost: a single dongle costs at least $10
- Distribution: of a dongle with software over the Internet is impractical

The dongle was once the protection technology of choice. However, there use has fallen out of favor. From a technical perspective, the dongle suffers from a major weakness which is that the attack point is clearly defined [17]. The interface to the device is a hardware interface which means that the signals passing over the interface must conform to the hardware standards. This gives the attacker an analysis advantage.

3.2.2 Tamperproof CPUs

Tamperproof CPUs aid in piracy prevention by providing a secure context and secure data storage. By executing the software in a secure environment the pirate is unable to gain access to the software. This technique prevents the attacker from observing the behavior of the software which means he is unable to identify portions of the software to remove. The obvious drawback to this technique is the additional cost of requiring all users to have tamperproof hardware. Lie [18] proposes one such technique in which standard hardware is modified so that encrypted code can be securely executed. To accomplish this, each "XOM" chip contains a different decryption key. To execute the encrypted code the processor enters XOM mode. The instructions are then decrypted and verified in the XOM Instruction Decryption Unit. The special Decryption Unit is only used in XOM mode so as to minimize the overhead incurred due to the hardware modifications. Prior to the processor switching from secure to normal operation mode the architectural state is secured. In this process the registers and cache are cleared and all pending writes are completed [1]. This prevents a user from waiting until the XOM mode has completed to obtain information which could reveal the encrypted instructions. Such architectural support makes it possible to maintain algorithmic secrets. Additionally, because each XOM chip uses a different decryption key it is possible to prevent execution of unauthorized copies of the software.

3.2.3 Smart Cards

Smart cards are used in many contexts to securely store data. For example, smart cards store cryptography keys for use in authentication systems and channel authorizations for use in broadcast television systems. Traditional smart card consists of an 8 bit microprocessor with ROM, EEPROM and RAM on a single chip with serial input and output. The EEPROM is used to store the secure information. Erasing this data requires a relatively high voltage, however, if the attacker can prevent the voltage from reaching the EEPROM the information will remain [9]. Early smart cards received their voltage from the host. Attackers were able to make use of this design to attack pay TV systems which used smart cards to store subscription information. In the attack, all channels were initially enabled. Prior to canceling the service, the attacker would cover the voltage contact using something as simple as tape. The voltage sent to the card never reached the

EEPROM, allowing the attacker to continue to use the service without paying. The next generation of smart cards raised the cost of attack, but still did not make it impossible. The design of these smart cards changed the source of the voltage used to reprogram the EEPROM. Attacks on this type of card can be carried out using a microscope and a laser. Anderson and Kuhn [1] describe a variety of attacks on smart cards along with the associated costs.

3.3 Software Based Techniques

Software solutions are concerned with making a program secure against reverse engineering and modification. This can be achieved through a number of different methods. Such as code obfuscation, software tampers proofing, software watermarks and software birthmarks. These methods will be discussed later in the following section. This solution provides a number of advantages over strictly hardware based techniques.

1. The protection is cheaper to implement due to the lack of special purpose hardware. Many of the currently proposed hardware based solutions have been easily attack. Some of the attacks require specialized equipment, but many of the side channel attacks are relatively cheap. For example, the protection provided by some smartcards can be defeated by shining a common light-bulb on the card [10]. Many hardware based solutions can be difficult to deploy. It can take many years for users to upgrade their machines to ones which contain tamperproof CPUs and once the protection mechanism has been defeated it cannot be fixed without upgrading the hardware again. The software based solution take a different approach than hardware based techniques because it is generally believed that given enough time a determined adversary will be able to defeat any protection mechanism. The goal instead is to develop techniques which require enough time, effort, and resources to break such that it is less costly for the attacker to simply rewrite the software or purchase legal copies. Thus, the techniques can be used to extend the period in which no pirated copies exist, increasing the revenue for the software producers [18]. This is especially useful for products such as video games which often have a short shelf life. Additionally, software based techniques can be used in conjunction with specialized hardware to increase the strength or to further protect the software once the hardware protection has been defeated.

3.3.1 Media-Based Protections

These have been around since the 1980's. The media, on which the copyrighted material is shipped, contains several specific features that allow verification of the authenticity of the media. In software distributions, the program checks if these features are present, whenever the program is executed. Since the 1980's much progress has been made in this

field and nowadays media based protection is the primary copy protection used in the gaming industry. Media-based protections range from specific bad sector on floppy disk, to advance techniques for protection of executables using byte code and cryptography on DVDs [9]

3.3.2 Serial-Based Protections

Using product serial numbers is one of the most common ways to verify the authenticity of legitimate users. The concept is to provide legitimate users with a serial, which is then checked by the program using a secret validation algorithm. This scheme is not exclusively used for online distributions. In fact, it was originally used in over-the-counter software. During installation of the software, the installer asks the user to insert the serial, if it is invalid the installation process terminates. Usually such a serial is printed on something bundled with the software [19]. In applications that can be registered online, the serial can be of a specific structure and use above described scheme To register an application, the user then contacts the author by sending him for his name and the author provides the user with a key, created on the basis of the user parameters.
This serial was generated using the vendor private key-generating algorithm. When the user enters his parameters and the key in the software registration box, the program calculates the key by running the user parameters through the built-in key generator and then compares the entered key with the one calculated in the background. When these two values match, the registration is successful. It should be mentioned that this protection is flexible and user-friendly, but has an inherent security risk, because the verification process includes generating the correct key on the end-users machine. Another weakness of serial-based protections is that there is no mechanism that prevents a same key to be used on different software installations that means allowing users to share keys [4].

3.3.3 Challenge Response

The challenge response mechanism is a well-known authentication protocol typically used for authenticating specific user or computer in a networking environment. The mechanism has also been applied as an improvement to the serial number protection scheme. The idea is that during installation of the application the end-user has to enter a registration number, comparable to that of the original scheme. The difference is that instead of just running this number through a verification algorithm, the installation program composes a unique challenge made up of the user supplied number and a unique machine identifier. This challenge is to be sent to the software vendor, who verifies that the serial number is legitimate. Following verification, the vendor responds with a key that is fed into the target program, where it is checked to be mathematically correct. While being slightly less flexible due to the requirement of network access during

registration, this approach is definitely a step up from the conventional serial number scheme, since serials cannot be used unchecked by pirates [20].

3.3.4 Service Model

In this scheme the software runs on servers maintained by the vendor of the software, the user has to be permanently connected to the Internet in order to use the program. While this scheme provides excellent protection, the requirement of permanent connectivity is a serious drawback nowadays mainly for security reasons. This makes the scheme, in its strictest form, not suitable for a variety of programs. However, a less stringent variant of the software as a service-model has successfully been used by several update-reliant programs for example, virus scanners and other security related software. In that case a user must authenticate him to the vendor servers in order to obtain the updates [5].

3.3.5 Digital Rights Management

Digital Rights Management (DRM) scheme is a technique that tries to control the flow of digital copyrighted material [14]. DRM is a developing branch of anti-piracy schemes that focus on controlling the flow of copyrighted media files. This scheme is relies on cryptography algorithms in which the decryption key should remain hidden to illegitimate users. Since the key is always required to enjoy the protected content, the main issue for DRM scheme is how to hide the key from the users on an entrusted and open system. The actual security code that performs decryption is not present in the media files themselves, but in the player that is used.

3.3.6 Code Obfuscation

Code obfuscation is a technique developed to aid in the prevention of malicious reverse engineering. This scheme is to attempt to make the attacker job understand an application infeasible by protecting primarily against static analysis [21]. Obfuscation gains its strength by combining a number of heuristics and algorithms which are designed to hide the function of the machine language code. This gives the attacker ability to gain a high level understanding of program flow. The high level understanding is useful in extracting critical algorithms that are used in applications the attacker may be developing. A high level of understanding is also useful when the attacker wishes to modify the application for their own gain for example, disabling copy protection on digital rights management programs to allow distribution of copyrighted material. There are three general classes of obfuscations:

- Layout obfuscation: alter the information that is unnecessary to the execution of the application such as identifier names and source code formatting.

- Data obfuscation: alter the data structures used by the program. For example, a two dimensional array could be folded into a one dimensional array.

- Control flow obfuscation: are used to disguise the true control flow of the application, for example by inserting dead or irrelevant code, converting a reducible flow graph into a non-reducible graph, in-lining methods, merging methods and transforming loops using techniques such as loop unrolling.

The most common and simplest obfuscation is name obfuscation. The basic idea is to rename the identifiers in the program to meaningless name. For example, the method getKey () could be rename to a (). Tyma [21] describes a technique in which method overloading is used to generate a few unique names as possible. For example, the methods foo and bar can be renamed to a. However, foobar must have a difficult name than bar since they have the same signature. Additionally, string cannot be renamed since it overrides java.lang, Object.toString. The same idea can be applied to fields and classes. When using renaming care must be taken with classes that are loaded by name and with fields and methods that are accessed using reflections. Code obfuscation has many interesting applications. In addition to rendering applications more difficult to understood and reverse engineer, it can be used to protect watermarked programs from a collusive attack. In this attack, an adversary obtains several differently watermarked programs and compares them to identify the watermark. Through the use obfuscation, differently fingerprinted programs can make the programs differ everywhere instead of only where the watermark was embedded. Obfuscation can also be used to perform a malicious attack against software watermarks, transforming the code such that the mark is unrecoverable.

3.3.7 Software Tamper proofing

Code obfuscation is used to hide a secret while tamper proofing is used to protect the secret form alteration. For example, many programs contain license checks that prevent the user from using the software after a specific date. An attacker will attempt to locate and disable the check in order to enjoy the software for free. To prevent such attacks a software developer may tamperproof the license check such that if it is altered the program will no longer function properly. A tamper proofing technique performs two duties [10]. The tamper proofing mechanism must detect that the software has been altered. Once detection has occurred, the mechanism must cause the program to fail.

For the tamper proofing to be successful, the software failure must be stealthy and not alert the attacker to the location of the failure inducing code. This can be accomplished by separating the detection and response mechanism in both space and time. The first non-trivial tamper proofing algorithm was published by Aucsmith [11]. The key to the algorithm are the Integrity Verification Kernel (IVK). These are small units of code that are responsible for performing critical blocks of code all of equal size. 2 N program

functions. Each IVK contains Half of these blocks are located in upper memory and the other half in lower memory. With the exception of the initial block each IVK block is encrypted. The blocks are executed in a pseudo random order determined by a key, beginning with the initial block. Once the code of the initial block has been executed the decrypt and jump function is performed. The function XOR operation means that each block in upper memory with a block in lower memory. The result of this operation is that at least one block in the lower memory has been decrypted and execution resumes at that block code section. This process continues, alternating between upper and lower memory blocks. Within each cell an accumulation product, sum of the hash function of all executed blocks) is checked to verify that the previous cells were executed correctly and in the correct order. This step occurs just prior to the decrypt and jump function and is responsible for verifying the integrity of the program. If a problem is detected in this Step appropriate action is taken which will eventually cause the program to fail. A second tamper proofing technique was proposed by Chang and Atallah [12] and implemented for Win31 executable. In this algorithm tamper protection is achieved by inserting a network of guards into the program. The network establishes a check and balance system by assigning different tasks to the guards. For example, one guard may checksum a section of code while another repairs it or two guards may check the integrity of each other. Through this network of guards the algorithm is able to verify if a section of code has been tampered with, These types of tamper proofing techniques work well for binary executable but are difficult to implement for type-safe distribution formats such as Java byte code. While it is possible to encrypt a Java class file and use a special class loader to load and decrypt it, it is impossible to do this in a stealthy way. It is always possible for an adversary to intercept the decrypted byte codes at the point where they are handed off to the Java Virtual Machine (JVM) for execution.

3.3.8 Software Water marking

Software water marking is used to embed a unique identifier in the program. Piracy is confirmed by proving the program contains the watermark. Watermarking can be used in one of two ways [13]. If each legal copy of the program is watermarked with the same identifier then the watermark is used as a proof of authorship. This type of mark is useful in cases where a module has been stolen and incorporated into another company application. By detecting the authorship mark the original creator is able to prove their software was stolen. A watermark can also be used to trace the source of the illegal distribution [14]. In this case each legal copy of the program contains a unique identifier, the fingerprint mark [15], which is linked to the original purchaser. If an illegal copy is identified the watermark will uniquely identify the guilty pirate. This technique is commonly referred to as fingerprinting.

3.3.9 Software Birth marking

A software birthmark is a unique characteristic, or set of characteristics, that a program possesses and which can be used to identify the program. Both have similar birthmarks q and p the general idea is that if two programs then it is highly likely that one is a partial or modified copy of the other. Just like software watermarks, software birthmarks are used to detect software theft. However, the techniques differ in two important ways. First, it is often necessary to add code to the application in order to embed a watermark. In the case of a birthmark, additional code is never needed. Instead a birthmark relies on an inherent characteristic of the application to show that one program is a copy of another. Secondly, a birthmark cannot be used to prove authorship or identify the source of an illegal redistribution. Rather, a birthmark can confirm that one program is contained in or is a partial copy of another. A strong birthmark will be able to provide evidence of software theft even when code transformations have been applied to the code by a malicious adversary [16].

References

[1] Felten E., "Understanding trusted computing: will its benefits outweigh its drawbacks", IEEE Security and Privacy, 1(03):60–61,2003

[2] Tuomas Aura, Dieter Gollmann, "Software License Management with Smart Cards", USENIX Workshop on Smartcard Technology Chicago, Illinois, USA, May 10–11, 1999.

[3] International Planning and Research Corporation, First Annual BSA and IDC Global Software Piracy Study, July 2007.

[4] Darren Bibby, John Gantz, Amie White, "The Impact of Software Piracy and License Misuse on the Channel", IDC, June 2008.

[5] Tavani H., "Ethics and Technology", Ethical Issues in an Age of Information and ommunication Technology, John Wiley and sons, Inc., 2004.

[6] Business Software Alliance, "The Economic Benefits of Lowering PC Software Piracy", January 2008.

[7] Digital millennium copyright act of 1998. Publication, No. 105-304, 112 Stat. 2860, October 18, 1998.

[8] Electronic Frontier Foundation, Unintended consequences: Five years .www.eff.orgunder the dmca, September 14, 2003.

[9] Vaddadi P. Chandu, Karandeep Singh, Ravi Baskaran, "A Model for Prevention of Software Piracy through Secure Distribution", , Advances in Computer and Information Sciences and Engineering Springer Netherlands, 2008..

[10] Chow S., Eisen P., Johnson H., and Van Oorschot, "White-cryptography and a DES implementation", Selected Areas in Cryptography, LNCS, 2595:250–270, 2003.

[11] Aucsmith D., Tamper resistant software: An implementation, in Information Hiding, First International Workshop, pages 317-333, Cambridge, UK, May 1996, Springer-Verlag, Lecture note in computer science, Volume 1174.

[12] Chang H. and Attallah, M., "Protecting Software Code by Guards", the First International Workshop on Security and Piracy in Digital Rights Management, ACM, pp. 160-175, 2001.

[13] Nagra J., Thomborson C., Collberg C., "A functional taxonomy for software watermarking", In M.J. Oudshoorn, editor, Twenty-Fifth, Australasian Computer Science Conference (ACSC1001), ACS, 2001.

[14] Anderson R., Kuth M., "Tamper resistance - a cautionary note", In USENLX Workshop on Electronic Commerce, pages 1-11, 1996.

[15] Collberg C. and Thomborson C., "Software watermarking: Models and dynamic embeddings", In Principles of Programming Languages, pages 311–314, 1999.

[16] Venkatesan R., Vazirani V., Sinha S., "A graph theoretic approach to software watermarking", Information Hiding, LNCS, 2137:157–168,2001.

[17] Software piracy protection device, Document Type and Number: United States Patent Application 20060185010, 2006, USA.

[18] ElGamal T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transaction on Information Theory, 1985. Eric.

[19] Olga Gelbart, Bhagirath Narahari, Rahul Simha, "A Secure Program Execution Environment Tool Using Code Integrity Checking", The George Washington University, Washington, DC, USA, Journal of High Speed Networks, 15, pp. 21-31, 2006.

[20] Protection of software using a challenge-response protocol embedded in the software, Document Type and Number: United States Patent 6651169, 2006.

[21] Tyma P., "Method for renaming identifiers of a computer program", US patent 6,102,966, 2000.

