Middle East University for Graduate Studies

Faculty of Information Technology

Precluding Software Piracy Using Asymmetric Cryptography

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Information System

By

Zeana Sattar Jabbar

Supervisor

Professor Mohammad A. Al-Fayoumi (PhD)

Dean of Information Technology Faculty

Middle East University for Graduate Studies

Amman - Jordan

Fabuary, 2009

جامعة الشرق الاوسط للدراسات العليا

نموذج تفويض

انا زينة ستار جبار افوض جامعة الشرق الاوسط للدراسات العليا بتزويد نسخ من رسالتي للمكتبات او الهيئات او الافراد عند طلبها

التوقيع:

التاريخ:

Middle East University for Graduate Studies

Authorization Form

I am Zeana Sattar Jabbar, authorize the Middle East University for Graduate Studies to supply copies of my Thesis to libraries or establishments or individuals on request.

Signature:

Date:

Committee Decision

This Thesis "**Precluding Software Piracy Using Asymmetric Cryptography**" was successfully defended and proved on January 20th 2009.

Examination Committee Signature	**Signature**
Professor Mohammad A. Al-Fayoumi (PhD) Department of Computer Information Systems Faculty of Information Technology Middle East University for Graduate Studies	
Professor Asim Abdel Rahman El Sheikh Department of Computer Information Systems Faculty of Information Technology Arab Academy for Finance and Banking	
Professor Rousam Department of Computer Information Systems Faculty of Information Technology Middle East University for Graduate Studies	
Dr. Hasan Al-Sakran**,** Department of Management of Information Systems College of Information Technology Yarmouk University, Jordan-Amman	

Dedication

This is dedicated to my family, for their encouragement and love.

Acknowledgments

First of all, I would like to thank God for his graces and for his guidance.

I would like to express my sincere appreciation to Professor Mohammad Al-Fayoumi for his guidance, assistance, scientific hints that lightened my road during my writing this Thesis.

I would further like to acknowledge all of the Information Technology faculty members at the Middle East University for Graduate Studies, for helping and encouraging my efforts especially at the beginning of the thesis.

Above all, I would like to especially thank my parents for supporting me during the time I was writing this thesis. Without them nothing of this thesis would have been possible.

List of Figures

Figure 3.1 shows the entire flowchart of the proposed scheme 38
Figure 3.2 shows the code Definition 45
Figure 3.3 Illustrate the phases of the proposed scheme 48

List of Tables

Table 3.1 Valid list of Copy IDs structure	40
Table 3.2 Products Structure	40
Table 3.3 Activations Structure	43
Table 3.4 Customer Structure	44
Table 3.5 Entity Relationship Model	47
Table 4.1 shows the steps involved in the computation of $5^{596} \bmod 1234 = 1013$	54
Table 4.2 shows the steps of algorithm 4.4 with input $a = 4864$ and $b = 3458$	57

الملخص

تعاني معظم موسسات العالم التجارية من مشكلة قرصنة البرمجيات . وكنتيجة حتمية تم تطوير العديد من الانظمة المتوفرة في الاسواق التجارية التي تعالج هذه المشكلة ولكن في الحقيقة ان معظم هذه الانظمة لا تقدم حلا مناسبا.

وبعد استعراض طبيعة النظم الحالية وجد المولف ان هذه النظم لم تاخذ بنظر الاعتبار المواصفات القياسية العالمية لمعالجة هذه المشكلة . وبذلك فانه من الصعوبة على هذه النظم منع او ايقاف القرصنة. لذا فان الغرض من هذه الرسالة هو تطوير نظام جديد يحمل خصائص المواصفات القياسية العالمية ليكون قادرا على منع القرصنة في اي بلد من خلال انتفاعه من خدمات الانترنيت والشبكة العنكبوتية ومن خلال استخدام احدى طرق التشفيرذات المفتاح العام الحتمية وبالاسم طريقة الجمل ومن خلال استخدام اسلوب اثبات الشخصية بالمعرفة الصفرية لضمان دخول المستخدمين للنظام بشكل صحيح وكذلك استخدام الرقم العالمي القياسي للنسخ لتسهيل العديد من هذه المعوقات .

النظام المقترح هو نظام متحرك وقابل للتوسع وكفوء ويدعي المولف من انه نظام اكثر قبولا مقارنة بالنظم المتوفرة في الاسواق . لذا فان المولف يوصي بتطبيق النظام من خلال دمج التطبيق البرمجي مع الارقام التسلسلية للمكونات المادية لتوليد هوية تعريفية جاهزة احادية . هذه الهوية ترسل الى المصنع للتحقق من صحه التطبيق وللتاكد من ان المنتج سوف لن يستخدم في تنصيبات متعددة . لذا فان المولف يوصي في تطبيق برنامج قياسي لجعل كل هذه الاساليب في برنامج واحد مطور .

النتائج المستخلصة ترسم صورة من ان النظام الجديد الذي يحمل العنوان:
"Precluding Software Piracy Using Asymmetric Cryptography"
يمكن ان يساعد بشكل كبير في عمل الموسسات التجارية التي تعاني من القرصنة.

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.

After reviewing the nature of existing systems the author found that these systems do not take in their considerations the international standard specification for treating this problem. Therefore, it is difficult for these systems to prevent or stop the piracy. Thus the purpose of this thesis is to develop a new scheme carrying the characteristics of international standard specifications in order to be able to prevent the piracy in any country by utilizing the Internet and Web services, and by using one from the deterministic public key encryption scheme namely Elgamal scheme and by using the zero knowledge proof of identity technique to grantee the users access to the scheme correctly and Also using the International standard copy number, to ease many of these difficulties.

The proposed scheme is dynamic, scalable and efficient and it claimed to be more acceptable compared with the already existed schemes on the market. Thus, the author recommends implementing a scheme where a software application merges hardware serial numbers to generate a unique Installation ID. This Installation ID is send to the manufacturer to verify the authenticity of the application and to ensure that the product is not being used for multiple installations. So, the author recommends implementing a standard program to make all these techniques on any developed program.

Results are given from which the conclusion is drawn that develop a new scheme entitled "Precluding Software Piracy Using Asymmetric Cryptography" can help significantly in the work of trade organizations that suffering from piracy.

Contents

Dedication IV
Acknowledgment V
List of Figures VI
List of Tables VII
Abstract in Arabic VIII
Abstract IX

Chapter One: Software Piracy
1. Introduction 1
    1.1 Problem Definition 3
    1.2 Objectives 3
    1.3 Motivations 4
    1.4 Significance 4
    1.5 Thesis Contrition 5
    1.6 Thesis Delimitations 5
    1.7 Thesis Organization 5
2. Related Work 6

Chapter Two: Software Piracy Solutions
1. Introduction 10
2. Deterrent Solutions 11
    2.1 United States Copyright Law 11
    2.2 Digital Millennium Copyright Act 12
    2.3 United States Patent Law 13
    2.4 United States Trademark Law 13
    2.5 United States Trade Secret Law 14
3. Preventive Solutions 14
    3.1 Audits 14
    3.2 Hardware Based Techniques 15
        3.2.1 Dongles 15
        3.2.2 Tamperproof CPUs 15
        3.2.3 Smart Cards 16
    3.3 Software Based Techniques 17
        3.3.1 Media-Based Protections 17
        3.3.2 Serial-Based Protections 18
        3.3.3 Challenge Response 18
        3.3.4 Service Model 19

3.3.5 Digital Rights Management 19
3.3.6 Code Obfuscation 19
3.3.7 Software Tamper proofing 21
3.3.8 Software Water marking 22
3.3.9 Software Birth marking 23

Chapter Three: The Proposed Scheme
3.1 Introduction 24
3.2 Algorithms Used 24
3.2.1 ElGamal Public Key Encryption Scheme 24
3.2.2 Data Encryption Standard (DES) 25
3.2.3 International Standard Copy Number 26
3.2.3.1 Check Digit 27
3.2.3.2 Check Digit Calculation 28
3.2.4 Zero Knowledge Identification 29
3.2.4.1 Zero Knowledge Concepts 30
3.2.4.2 Interactive Proof System and Zero Knowledge Protocol 30
3.2.4.3 Zero Knowledge versus other Asymmetric Protocols 32
3.2.4.4 The Proposed Protocol 32
3.2.4.5 Example 33
3.2.4.6 Security of identification Protocol 34
3.3 Code definitions 35
3.3.1 Machine ID 36
3.3.2 Installation ID 36
3.3.3 Copy ID 36
3.3.3.1 Products 37
3.3.3.2 Copy ID conditions 38
2.3.4 Internet Protocol address 38
2.3.5 Activation Code 39
3.4 Obtain the Hardware Information 42
3.5 Authentication 43
3.6 Customer Tracking System 43
3.6.1 Database 43
3.6.2 Website 44
3.6.3 Web service 44
3.7 The phases of the proposed scheme 45

Chapter Four: Tools Used
4.1 Introduction 48
4.2 Selecting a prime p and generator of Z_p^* 48
4.3 Miller-Rabin Test 50
4.3.1 Example 50
4.4 Square and Multiply Algorithm 51
4.5 Algorithm 4.1.3 to find a generator a of Z_p^* 52
4.5.1 Example 52
4.6 Greater common Divisor (gcd) 52
4.6.1 Example 53
4.7 Inverse 53
4.7.1 Example 54

Chapter Five: Conclusions and Future Work
5.1 Introduction 55
5.2 Future Work 55

References 57
Appendix A 62

Chapter One: Software Piracy

1. Introduction

Software Piracy is the unauthorized copying, reproduction, usage, or manufacturing of packaged software. Piracy can run the gamut from unauthorized copying or downloading of software or purchasing software copied illegally, to corporate misuse of volume licenses, deploying more software than paid for [37].

According to reports on software piracy [15], no existing protective measures have been able to meet the major challenge posed by software piracy. Among the approaches that have been explored in recently history to address the problem of software piracy are legal, ethical and technical means. Permission to make digital or hard copies of all or part of this thesis for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission or a fee. Legal means are based on the fear of consequences of violating piracy laws.

But while in most software piracy cases the legal of means are available, prosecution on a case by case basis is not economically in viable. Furthermore, it is conceived as bad publicity and can take a long time. Ethical measures relate to making software piracy morally unappealing. While the intentions are laudable, it takes effort and time to change the moral standards of a large group of people. The existing technical means almost all have a static nature of defense, in which a protection mechanism is built into the distributed software. Once this protection is broken no further steps can be taken to protect the intellectual property. And since any static protection is eventually broken, the existing techniques are not satisfactory at all [45].

To tackle this problem, this thesis presents an alternative technical protection scheme, whose strength is based on diversity. In the scheme the author presents, each installed copy of a program is unique. More precisely, each installed copy differs enough from all other installed copies to guarantee that successful attacks on its embedded copyright protection mechanism cannot be generalized successfully to other installed copies. Furthermore, the proposed scheme includes software updates to migrate from a static nature of defense to a more dynamic one. In particular, software updates in the proposed scheme are crafted to ensure that they work for one, and only one, installed copy. If updates are no longer provided for installed copies that are known to be illegitimate, a pirate needs to break through a new line of defense with every critical update. An additional advantage of the proposed scheme is a fine grained level of control over the distributed copies. This follows from the fact that a software provider in the proposed scheme can enable the installation of a copy on an arbitrary number of machines, or even tolerate an arbitrary level of software piracy. It will refer to the latter as piracy discrimination.

The worldwide revenue of business-based personal computer applications was $21.6 billion in 2003, but the global revenue lost due to piracy in the business application software market which was calculated at $12 billion in the same year [10]. In 2008 study by the U.S. research firm IDC showed that 13 percent reduction in software piracy in the EU (European Union) could boost the IT industry's growth rate from the current 30 percent to 38 percent by 2009 [23]. Because piracy also depresses demand for software design, customization and support, the study estimated that 13 percent reduction could add about $350 billion to economies worldwide and $72 billion in tax revenues [22].

For software publishers, a less expensive method of copy protection is to write the software so that it needs certain evidence from the user that they have actually purchased the software. They tend to generate methods to allow only authorized users to employ the programs. It is estimated that $29 billion of the total $80 billion of software installed on computers was actually installed illegally. However, in 2007 about 82 percent of software used in Romania is said to be pirated. Piracy rates in 2007 ranged from 92% in Vietnam and China to 22% in the United States with a global piracy rate of 36% [48]. Also, it was found that losses due to piracy in the Middle East region equals to 1,997 million dollars in 2007 [48]. The Business Software Alliance (BSA) defines four common types of software piracy:

Piracy occurs when a user reproduces copies of software without authorization. It can manifest itself in one of the following forms:

1. A user obtains a single licensed copy and uses it to install the software on multiple computers.
    - The disk used to install the software is duplicated and then distributed.
    - Within a commercial environment, employees use software with an academic license.
    - Network piracy occurs when a program is installed on a network and is simultaneously used by more people than the license entitles.
2. Internet piracy occurs when illegal copies of software are made available on the Internet either a free of charge or for a fee. Examples of such sites include:
    - Sites which make software available for free or by exchanging uploaded programs.
    - Auction sites that offer illegal software.

- Peer-to-peer networks which enable the transfer of illegal software.

3. Hard disk loading occurs when illegal software is installed on a new computer and sold. This activity often occurs when a business is trying to cut costs to make their products more attractive.
4. Software piracy occurs when copyrighted material is illegally duplicated and sold with the intent to it is original.

1.1 Problem Definition

Software is Intellectual Property; it should be protected from unauthorized users in order to ensure that the existing revenue runs. Software piracy continues to grow globally because it is cheap and easy to copy. The effects of this growth are devastating: not only does software piracy reduce revenues, it also results in less research and development, and in less investment in marketing and channel development.

It is important to minimize piracy rates as much as possible. Since the legal methods fail to prevent software piracy, it is necessary to protect software from pirates using technical and mathematical methods. In this thesis, the author proposes a new scheme that would help prevent software piracy using ElGamal scheme.

1.2 Objectives

The objectives of this thesis are as follows:

1. To generate a secured and efficient method that prevents unauthorized users to pirate.

2. To develop a secured way to monitor and track the users when they follow certain software.
3. To provide an easy way to the software vendors to protect their software by implementing a generic application of this technique.

1.3 Motivations

The motivations of this scheme are as follows:

1. To reach great values and results.
2. Due to the weak mechanisms that characterized the current schemes such as having some points of failure or they have performance penalty. The suggested scheme addresses several aspects which makes it a uniquely effective method.
3. To obtain remarkable results using new tools such as .Net frame work, and other methods such as ElGamal scheme, Euclidean inverse method, and zero knowledge protocol. Without these methods we cannot make this scheme work appropriately.
4. Due to the damage that piracy caused to the local markets, no scientific scheme is developed to study the local pirated problem.
5. To reach the results that is hard to reach in the past. This scheme is possible to programming using computer with a 3GHz Intel Pentium two processors with the use of programming language such as C#.

1.4 Significance

This thesis lives to serve both the software developer and the software user in the following points:

1. The developers will be able to develop without feeling threatened by the ghost of piracy, which will give them more time to concentrate on the development process rather than the protection process.

2. The users will benefit from the provided technical support which will lead to the evolution of the program.
3. Previous studies concentrated on one aspect and neglected other problems while this thesis tries to examine several aspects and conditions.
4. The administrators will be able to prevent piracy in these firms. Only the authorized persons will be able to use their applications in the authorized places only.

1.5 Thesis Contribution

The contributions of the proposed scheme can be characterized under a number of properties.

1. The proposed scheme is more efficient since it provides no execution time penalty on the protected program.
2. The proposed scheme is dynamic. It can be easily applied on any .net based software without the need for its source code.
3. The proposed scheme is scalable since it is based on a method that does not bind the user to a specific version of the operating system
4. The proposed scheme serves all kind of users. Home users with no network connection, enterprise users who should execute their applications only from a specific location and other users who have mixed conditions.

1.6 Thesis Delimitations

The delimitations of this thesis are as follows:

1. As long the size of computer address is 32 bytes, it must fill all these bytes by a serial numbers of four of the computer components. For example the serial numbers of hard disc, mother board, key board,

and CPU. It means that each serial number occupied a limited sub address of 8 bytes.

2. In order to generate a secure piracy scheme we must choose in ElGamal scheme a large random prime number p by which we mean one with equally around 1024 bits, such that $p-1$ is divisible by another medium private q of around 160 bits.
3. After manufacturing the scheme patch we cannot make any change on the scheme for example to change the keyboard unless we must contact again with the manufacturer and inform it about the change in order to produce another patch.

1.7 Thesis Organization

The organization of this thesis is as follows:

1. The rest of chapter one describe the related work
2. Chapter two gives theoretical analysis about the piracy and its prevention techniques.
3. Chapter three describes the proposed scheme. It explains its parts and how it works.
4. Chapter four describes the analysis efficiency and it discusses the obtained results from the scheme.
5. Chapter five describes the conclusions and future work regarding the proposed scheme.

2. Related Work

This section provides an overview of related work and identifies the fundamental weaknesses of the existing technical means for software piracy prevention.

In 1989 (Chandra, Comerford, White) [3] proposed a scheme entitled "Software protection using single key cryptosystem". This scheme provides a software protection mechanism which is based on the separation of the protected software from the right to execute the software. Protected software can only be executed on composite computing systems in which a physically and logically secured coprocessor is associated with a host computer. This scheme is broken down into using encryption algorithm. This scheme provides secure execution, where only the authorized user can run the program. It is commonly known as "dongle based method". This method bounds the user to the coprocessor. It is irreplaceable unit, and proved to be not compatible with new operating systems. The users of this scheme had always suffered when they tried to upgrade their systems. Compare with the proposed scheme this scheme is symmetric scheme while the proposed scheme is asymmetric scheme.

In 1995 (Barber, Woodward, Burkley, Rehme, Jackson, Young) [25] proposed a scheme entitled "System for controlling the number of concurrent copies of a program in a network based on number of available licenses". This scheme allowed one copy of a computer program to be available for use at each of a plurality of nodes of a network. If a valid license file at a local node contains an unexpired available license, a license manager at the local node permits the computer program to be executed at the requesting local node. If no such license is available in a valid license file at such local node, the license manager searches the other nodes for a valid license file containing an unexpired, available license. If an unexpired available license is located in a valid license file at a second or remote node, the license manager transfers such license to the local node, and assigns and encrypts a unique identification to such transferred license. The original record of the transferred license is modified by erasing it from the license file at the remote node so that the transferred license is no longer available there. The number of copies of the computer program that are authorized for execution simultaneously on the network is thus limited to the number of licenses that have been loaded into the license files on the network. This scheme is not applicable in non-network based systems, or those systems which do not always have network connection.

In 2000 (Schmid, Hill, Ghosh) [28] proposed a scheme entitled “Preventing the Execution of Unauthorized Win32 Applications”. In this scheme they describe an approach and tool for providing administrative control over the execution of software on a Windows NT/2000 system. As a result, pirated, and malicious software executables can be prevented from running on corporate machines. The scheme provides the enterprise wide facility for controlling the software that is allowed to run on users’ machines. Bear in mind that because the control mechanism is based on running the software, it does not prevent unauthorized software installations it merely prevents their execution. This method tends to virtually prevent unauthorized user from executing the application. It only achieves the goals within a closed environment. This method cannot be applied on broadly used desktop computers.

In 2002 (Chang, Attallah) [7] proposed a scheme entitled “Protecting Software Codes by Guards”. In this scheme small pieces of codes (or guards) are inserted throughout the code during compilation. Each guard is responsible for check summing a particular piece of code. If tampering is detected, a special kind of repair guard or code is called. This approach is based on a distributed scheme, in which protection and tamper resistance of program code is achieved, not by a single security module, but by a network of smaller security units that work together in the program. This scheme does not by itself prevent unauthorized use of the program. It must be accompanied with another method to achieve that. This method is designed to prevent crackers from cracking the software. This method eliminates a single point to failure. But, it does not only increase code size, but also it is vulnerable to code analysis which can remove the guards before execution.

In 2006 (Gelbart, Narahari, Simha) [31] proposed a scheme entitled “A Secure Program Execution Environment tool using code integrity checking”. This scheme is for software integrity protection and authentication. The scheme architecture utilizes key components from the compilation process and operating system support to provide static verification of executables. Code integrity checking is performed by means of a hierarchical hashing scheme, which not only detects changes but also efficiently isolates them. This scheme provides a higher level of protection against code injection or modification than a simple chaining of the program blocks. As an additional benefit, it also provides forensic information in case of a verification failure by providing the user with information about which part of the program has been modified. The scheme is designed to function as part of the operating system kernel in order to provide a trusted computing system. This scheme combines concepts from compilers, operating systems and watermarking to provide code verification and authentication thereby preventing code tampering attacks. In addition, it provides forensic information to the user about what exact part of the code has been attacked. This scheme contributes to the creation of a trusted computing system. However, it is extremely complex, and bounds the user to a specific Linux based environment where the kernel is modified. It cannot be applied on commercial applications.

In 2007 (Vassiliadis, Bill, Fotopoulos, Vassilis) [46] suggested a scheme entitled "Software protection based on watermarking". This scheme is designed to create, use, and distribute digital content through e-commerce channels. As online corruption increases, new technical and business requirements are posed for protecting the software applications using watermarking, use of metadata, self-protection, and self-authentication.

This scheme is a review of the most important of these methods and analyzes their potential use in digital rights management systems. It focuses especially on watermarking and argues that it has a true potential in e-business because it is possible to embed and detect multiple watermarks to a single digital artifact without decreasing its quality. Compare with the proposed scheme this scheme is symmetric scheme while the proposed scheme is asymmetric scheme.

In 2008 (Holsapple, Iyengar, Haihao, Rao) [8] proposed a scheme entitled "Software Piracy based on Routine Activities Theory, Rational Choice Theory, and guardianship concepts Advances". This scheme is for Internet and other digital technologies that have opened up new channels and methods for online business. They have also led to a situation where the same channels can be abused and misused. One of these forms of technology abuse, which is becoming increasingly prevalent these days, is the piracy of digital content. This scheme introduces a relatively comprehensive and unified theoretical framework for studying and understanding a major aspect of digital piracy: namely, software piracy. This scheme identifies key parameters that can affect the incidence of software piracy. It applies the framework in conducting a systematic examination of 75 articles dealing with software piracy. The examination reveals that a considerable number of parameters have received little or no attention from software piracy researchers. In addition to suggesting research opportunities, the scheme furnishes a systematic approach for structuring the design of future research studies in the realm of software piracy. The insights furnished by this scheme contribute to future investigations of the software piracy phenomenon that are needed to avert the economic and social damage caused by software piracy. This scheme is not applicable in non-network based systems, or those systems which do not always have network connection. As long as this scheme relies on the operation of the finite state technique will finally be cracked.

Chapter Two: Software Piracy Solutions

2.1 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 [19]. 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 [41]. 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 2002, 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 [23].

2.2 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 [12]. Currently, four intellectual property rights laws can be applied in the protection of software. These laws include copyright, patent, trademark and trade secret.

2.2.1 United States Copyright Law

Under the 1978 United States Copyright Act, as described by Tavani [41], 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.

2.2.2 Digital Millennium Copyright Act

The Digital Millennium Copyright Act (DMCA) was signed into law October 28, 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 1201, circumvention of copyright protection systems, of the DMCA [13] 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 [16].

2.2.3 United States Patent Law

The United States Patent Law provides legal protection to individuals who create an invention or process [26]. 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.
2. 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 20,000 new software patents are issued each year despite considerable debate over the appropriateness of patenting software [41].

2.2.4 United States Trademark Law

Kizza [26] 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.

2.2.5 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 [26]. 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.

2.3 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.

2.3.1 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.

2.3.2 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 [22]. Such hardware is typically more cumbersome for the user and more expensive for the software vendor than software based techniques.

2.3.2.1 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 [29]. 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 [38]. 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.

2.3.2.2 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 [27] 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 [21]. 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.

2.3.2.3 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 micro processor 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 [42].

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.

2.3.3 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.
2. 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 [20].
3. 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.

2.3.3.1 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 [44].

2.3.3.2 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 [32].

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].

2.3.3.3 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 [33].

2.3.3.4 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].

2.3.3.5 Digital Rights Management

Digital Rights Management (DRM) scheme is a technique that tries to control the flow of digital copyrighted material [24]. 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.

2.3.3.6 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 [43]. 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 [43] 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.

2.3.3.7 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 [21].

- 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 [4]. 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 [7] and implemented for Win32 executables. 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 executables 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.

2.3.3.8 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 [30]. 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 [2]. In this case each legal copy of the program contains a unique identifier, the fingerprint mark [9], 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.

2.3.3.9 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 [47].

Chapter Three: The Proposed Scheme

3.1 Introduction

The problem of protecting software from illegal copying and redistribution has seen considerable attention in both recent research and updated legislation. The growing concern regarding software piracy can be attributed to a variety of factors such as rich distribution formats, which preserve much of the information from the source code, and the ease of sharing over the Internet. In previous years piracy was limited by the necessity to physically transfer a piece of software on a floppy disc or CDROM. With the increases in bandwidth physical transfer is no longer necessary.

Currently, there are a variety of techniques in use to try to prevent, discourage, and detect theft. The pros and cons of each must be weighed by the software vendors to decide if the protection afforded by the technology is worth the additional cost. Unfortunately, no single solution is currently strong enough to completely prevent piracy. However, through a combination of techniques, application developers can better protect their products. For many companies the goal is simply to protect the software long enough that a reasonable return on the investment can be obtained. Since current technologies are limited in their protection capabilities continued research into more robust techniques is necessary. Therefore, in this chapter, the author will propose a new scheme that is based on a combination of technique the serial number protection combined with ElGamal encryption scheme and other techniques to produce a robust scheme.

3.2 Algorithms Used

The proposed scheme requires three algorithms to be used in order to work two from these algorithms are developed these are International

Standard Copy Number and Zero Knowledge Identification. The required algorithms are as follows:

3.2.1 ElGamal Public Key Encryption Scheme

In 1976 Diffie and Hellmam [14] created the first revolutionary research in public key cryptography there was presented a new idea in cryptography and they challenged experts to generate cryptography algorithms that faced the requirements for public key cryptosystems. However, the first reaction to the challenge is introduced in 1978 by RSA [34]. The RSA scheme is a block cipher in which the original message and cipher message is composite modulus. n where $[0 \ldots n-1]$ are integer values in the interval The security of the RSA is based on the difficulty of finding the private given only the public key, namely the public d encryption exponent . The intractability of the e and the public encryption exponent n modulus RSA assumption forms its security. In other words, the RSA difficulty is that which n the RSA assumption is the difficulty of solving the integer modulus with an assistance of another q and p is a product of two large prime . The other reaction to this c and an integer cipher text e public key challenge is introduced in 1984 by ElGamal [17]. The ElGamal encryption algorithm is based on the discrete logarithm problem. The ElGamal encryption scheme is deterministic whereas the RSA is probabilities which unlike the RSA algorithm, in ElGamal there are some public parameters which can be shared by a number of users. There are called domain parameters. The ElGamal scheme is described in Appendix A:

3.2.2 Data Encryption Standard

The objective of this section is to illustrate the principles of modern conventional encryption. For this purpose, we briefly focus on the most widely used conventional encryption algorithms, the Data Encryption Standard (DES) [11]. Although numerous conventional encryption algorithms have been developed since the introduction of DES, it remains the most important such algorithm.

The differences between the DES and most conventional encryption algorithms with the public key encryption schemes such as RSA and Elgamal are as follows:

1. The structure of DES and other conventional encryption algorithms is very complex and cannot be explained as easily as RSA and Elgamal algorithms.
2. The DES and other conventional encryption algorithms are using single key, it means that they used the same key in both the encryption and decryption operations.
3. The DES and other conventional methods are using simple operation that is shifting operation with the OXR function. Compare with the public key schemes such as the RSA and ElGamal they used more solid operations.
4. Compare with the public key schemes, the DES is not secure enough since it is totally broken down in 1998 [39]. While the public key schemes are quite secure.

Therefore, the author decides to use Elgamal scheme to ensure that the scheme will be more secure and more reliable than DES scheme.

Accordingly, we can use a simplified version of DES. This version allows the reader to perform encryption and decryption by hand and gain a good understanding of the working of the algorithm details.

Simplified DES is an educational rather than a secure encryption algorithm. It has similar properties and structure to DES with much smaller parameters. It was developed by Professor Edward Schaefer of Santa Clara University [36].

3.2.3 International Standard Copy Number

In a good cryptography scheme, changing one bit in the cipher text changes enough bits in the corresponding message to make it unreadable. Therefore, it needs a way of detecting and correcting errors that could occur when cipher text is transmitted [35].

Many non cryptography situations also require error correction for example fax machine, computer hard drives, CD players, and anything that works with digitally represented data. Error correcting codes solve this problem.

The international standard copy number (ISCN) provides an example of an error detecting code. The ISCN is an identifying number assigned to virtually every software copy. A new edition receives its own ISCN. It serves to uniquely identify the software copy. Every ISCN for example has four parts:

1. country source code
2. publisher code
3. serial copy number
4. Check digit

For example, a total of 10 digits ISCN 03-87-95045-1 have the following:

The county code: in this case the third country source code, Australia. In this example the item recorded by two digits and will allow for a range (00 to 99).	03
The publisher code: This represents the publisher unit number. This example allows for a range of (00 to 99) to each publisher.	87

This means that there are 100 companies produce software in each country.	
The serial number: This represents the software serial number. This example allows for a range of (00001 to 99999) in each company software publication.	95045
This is the check digit which verifies the accuracy of the preceding combination of numbers. See section 1.3.1for check digit calculation.	1

Table below lists some country source code.

United Kingdom	00
United State of America	01
New Zealand	02
Australia	03
Canada	04
South Africa	05
Zimbabwe	06
France	07
Belgium	08
Switzerland	09
Germany	10
Austria	11
Japan	12
Pakistan	13

China	14
India	15
Saudi Arabia	16
Jordan	17
Spain	18
Egypt	19
Greece	20

3.2.3.1 Check Digit

In this section one from the important algorithm that is used throughout the scheme is described. This algorithm is the check digit.

Whenever many long identification numbers must be typed into a computer, the chances are high that typographical errors will be made. One way of reducing these errors is to include a check digit in any given number. If the number is entered incorrectly then the check digit will probably no longer correspond and an error will be detected.

The algorithm in the check digit should be designed to perform three basic functions which are as follows:

1. To read the serial number with or without check digit.
2. To produce the check digit for a serial number that does not have one.
3. To verify the check digit of a number that does not have one.

The algorithm is designed to impose a strict format on the numbers it reads. Assuming serial numbers are nine digits long plus a check digit ten digit in all as described above. The algorithm should accept an input value in one of two forms, with or without a check digit. For an input value containing exactly nine digits (for example 123456789) the numerical value is assigned and the absence of a check digit is the input number is detected. An input value with a check digit requires a hyphen between the last digit of the serial number and the check digit:

123456789-7

In this case the program receives the ten digit numerical value hyphen deleted. An invalid input will result in an error message and the prompt for another attempt, until an input ten digit number nine digits, hyphen and check digit is valid.

3.2.3.2 Check Digit Calculation

The check digit is computed by multiplying the leftmost ISCN digit by 10, the next digit by 9, and so on up to the ninth digit from the left, which is multiplied by 2. The products are then added, and the check digit is determined as the smallest integer that when added to this weighted sum will make it a multiple of 11. The check digit is therefore in the interval [0, 10]. If it happens to be 10, it is replaced by the Roman numeral X in order d_1 to make it a single symbol. If we denote the nine leftmost ISCN digits by is computed by first I (from left to right), then the ISCN d_9 through calculating the weighted sum:

$$T = (10d_1 + 9d_2 + 8d_3 + 7d_4 + 6d_5 + 5d_6 + 4d_7 + 3d_8 + 2d_9) \bmod 11$$

because the use of the mod and then $[0...10]$ is in the interval T Notice that . For example, given the nine digits 038795045, the two $I = 11 - T$ subtracting steps produce.

$T = (10*0+9*3+8*8+7*7+6*9+5*5+4*0+3*4+2*5) \bmod 11 = 241 \bmod 11 = 10$
yielding ISCN 03-87-95045-1 ISCN are assigned, printed, $I = 11 - 10 = 1$ and scanned and handled by both machine and humans, so errors can creep in. It is important to detect errors, but there is no need to automatically correct them by means of a sophisticated error correcting code. When the check digit indicates an error in the ISCN, a human can easily identify the error and correct it manually. Obviously, a single check digit cannot detect every error, but it is easy to show that the ISCN check digits can detect all the most common errors. The most common errors in an ISCN are a corrupted digit and two consecutive digits being transposed. It is easy to show that all these errors will be detected by the ISCN check digit.

To understand why single digit errors are always detected, the author try to will not d_i find cases where such an error will go unnoticed. An error in digit , this $\bmod 11$ is computed T. Since T be detected unless it affects the value of by 11 $10d_1 + 9d_2 + ... + 2d_9$ will happen if the corrupted digit changes the sum is prime and since the weights that 11 or by multiple of 11. However, since multiply the nine digits are relatively prime to 11, this cannot happen. A , $9*10$ by a small multiple of 10 up to T for example will change d_1 change in will d_2 and such multiple is never a multiple of 11. Similarity, a change in , which is never a multiple of 11, $9*9$ by a small multiple of 9 up to T change . Their d_5, and d_4 and so on. Now examine two consecutives digits, such as , but when transposed they $7d_4 + 6d_5$ is the sum T contribution to to remain unchanged, the difference of T. In order for $7d_5 + 6d_4$ contribute the two contributions should be a multiple of 11, but the difference and the difference of two digits is a digit in $d_5 - d_4$ or equivalently $d_4 - d_5$ is , it never equal 11. $-9,+9$ the interval

3.2.4 Zero Knowledge Identification

This section considers protocol specifically designed to achieve identification, which use asymmetric techniques but do not rely on digital signature or public key encryption and which avoid use of block ciphers, and timestamp. It is similar in some regards to the challenge-response protocol, but also based on the idea of interactive proof systems and zero knowledge proofs employing random numbers not only as challenges, but also as commitments to prevent cheating.

3.2.4.1 Zero Knowledge Concepts

The disadvantage of simple password protocols is that when a claimant called a prover in the context of zero knowledge protocol gives the A can thereafter impersonate the B her password. The verifier B verifier responds to A. Challenge response protocol improve on this: A prover secret in a time variant A challenge to demonstrate knowledge of B. This might B manner, providing information not directly reusable by nonetheless reveal some partial information about the claimant secret: an adversarial verifier might also be able to strategically select challenge to obtain response providing such information.

Zero knowledge protocol are designed to address these concerns, by allowing a prover to demonstrate knowledge of a secret while revealing no information whatsoever of use to the verifier in conveying this demonstration of knowledge to others. The point is that only a single bit of information need be conveyed namely, that the prover actually does know the secret. More generally, a zero knowledge protocol allows a proof of the truth of an assertion while conveying no information whatever about the assertion itself other than its actual truth. In this sense, a zero knowledge proof is similar to an answer obtained from a trusted oracle [40].

3.2.4.2 Interactive Proof System and Zero Knowledge Protocol

The zero knowledge protocol to be discussed are instances of interactive proof system, wherein a prover and verifier exchange multiple messages (challenge and response), typically dependent on random numbers, which they may keep secret. The prover objective is to convince the verifier the truth of an assertion for example claimed knowledge of a secret. The verifier either accepts or rejects the proof. The traditional mathematical notion of a proof is altered to an interactive game wherein proofs are probabilistic rather than absolute, a proof in this context need be correct only with bounded probability albeit possibly arbitrary close to 1. For this reason, an interactive proof is sometimes called a proof by protocol.

Interactive proofs used for identification may be formulated as proofs of it has B and attempts to convince s Possesses some secret A knowledge. by correctly responding to queries which require knowledge s knowledge of differs from proving that s to answer. Note that proving knowledge of s of exists, for example proving knowledge of the prime factors of s such is composite. n differs from proving that n

An interactive proof is said to be a proof of knowledge if it has both the properties of completeness and soundness. Completeness may be viewed as the customary requirement that a protocol functions properly given honest participants. However, a zero knowledge proof must satisfy three properties:

Completeness property: An interactive proof protocol is complete if given an honest prover and an honest verifier; the protocol succeeds with overwhelming probability that is the verifier accepts the prover claim. The definition of overwhelming depends on the application, but generally implies that the probability of failure is not of practical significance. This means if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.

Soundness property: An interactive proof protocol is sound if there exists with the following property: if a m an expected polynomial time algorithm can with non negligible probability A dishonest prover impersonating can be used to extract from m then B successfully execute the protocol with secret, which with A this prover knowledge essentially equivalent to overwhelming probability allows successful subsequent protocol executions. This means if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability. if the statement is true, no cheating verifier learns anything other than this fact. This is formalized by showing that every cheating verifier has some simulator that, given only the statement to be proven (and no access to the prover), can produce a transcript that "looks like" an interaction between the honest prover and the cheating verifier. However, there are techniques to decrease the soundness error to negligibly small values.

The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero knowledge. Research in zero knowledge proofs has been motivated by authentication systems where one party wants to prove its identity to a second party via some secret information (such as a password) but does not want the second party to learn anything about this secret. This is called a "zero knowledge proof of knowledge". However, a password is typically too small or insufficiently random to be used in many schemes for zero knowledge proofs of knowledge. A zero knowledge password proof is a special kind of zero knowledge proof of knowledge that addresses the limited size of passwords. Zero knowledge proofs are not proofs in the mathematical sense of the term because there is some small probability, the soundness error, that a cheating prover will be able to convince the verifier of a false statement. In other words, they are probabilistic rather than deterministic.

3.2.4.3 Zero Knowledge versus other Asymmetric Protocols

The following observations may be made regarding zero knowledge techniques, as compared with other public key techniques.

1. No degradation with usage: protocols proven to have the zero knowledge property do not suffer degradation of security with repeated use, and resist chosen text attacks. This is perhaps the most appealing practical feature of zero knowledge techniques.
2. Encryption avoided: many zero knowledge techniques avoid use of explicit encryption algorithms. This may offer political advantages for example with respect to export controls.
3. Efficiently: while some zero knowledge based techniques are extremely efficient protocols which formally have the zero knowledge property typically have higher communications and computational overheads than public key protocols which do not. The computational efficiency of the more practical zero knowledge based schemes arises from their nature as interactive proofs rather than their zero knowledge aspect.
4. Unproven assumptions: many zero knowledge protocols proofs themselves and rely on the same unproven assumptions as public

key techniques for example the intractability of factoring or quadratic residue.

5. Zero knowledge versus zero knowledge: although supported by prudent underlying principles, many techniques based on zero knowledge concepts fall short of formally being zero knowledge and formally sound in practice, due to parameter selection for reasons of efficiency, or for other technical reasons. In fact many such concepts are asymptotic and do not apply directly to practical protocols.

3.2.4.4 The Proposed Protocol

In cryptography, a zero knowledge proof or zero knowledge protocol is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement.

is assumed. T In the following protocol the existence of a trusted authority which equals the n The only purpose of the agency is to publish a modulus but to keep the primes themselves q and p product of two large prime secret. For a technical reason to be explained later, the primes are assumed the trusted authority may n. After publishing $3 \bmod 4$ to be congruent with $d_1 \ldots d_k$ numbers k consists of d_A secret identification A cease to exist. Entity with $e_1 \ldots e_k$ numbers k consists of e_A. His public identification $1 \le d_j < n$ with . The $e_j * d_j \equiv \pm 1 \bmod n$ satisfying one of the congruence e_j and each $1 \le e_j < n$ wants to convince A. Entity e_j and the n knows the public B verifier entity . The following four steps constitute one round of the d_A her that he knows A protocol. The number of rounds decreases the probability of entity cheating.

1. Entity A chooses random number r and computes the number $\pm r^2 \bmod n$ and tells one of them call it x to entity B
2. Entity B chooses a subset s of the set $\{1 \ldots k\}$ and tells it to entity A.

3. Entity A tells entity B the number $y = r * T_d \bmod n$ where T_d is the product of the numbers d_j such that j belongs to s.
4. Entity B verifies the condition $x \equiv \pm y^2 * T_e \bmod n$ where T_e is the product of the numbers j_e such that j belongs to s. Using one from the inverse methods (using algorithm 4.3 details in chapter 4) and Euclid greater common divisor ((gcd) (using algorithm 4.4 details in chapter 4). If it is not satisfied, entity B rejects. Otherwise, an eventual new round is begun.

3.2.4.5 Example

his identity in order to access a B wishes to prove to entity A Entity resource.

. The initial A may ask the following prove that you are entity B Entity published T authentication problem is fully solved by the trusted authority $n = 47 * 59 = 2773$. Then $q = 59$, and $p = 47$. Suppose that $n = p * q$ the modulus

in a secure n The trusted authority can distribute the identification modulus fashion, for example by hand or over encrypted and authenticated lines. but not its factorization $n = 2773$ gets from the trusted authority B Entity as they must keeping secret with the trusted authority and q and p that is no one must know about them. Suppose that the secret identification which consists of 6-tuple at random: d_j

$$d_1 = 1901, d_2 = 2114, d_3 = 1509, d_4 = 1400, d_5 = 2001, d_6 = 119$$

to consist of 6-tuple: e_j now will choose his public identification A Entity

$$e_1 = 81, e_2 = 2678, e_3 = 1207, e_4 = 1183, e_5 = 2681, e_6 = 2595$$

and moreover +1 appears $j = 1,\ldots,6$ Then the congruence will be satisfied for . Assume that entity $j = 2,6$ and -1 appears for $j = 1,3,4,5$ on the right side for the number B randomly and tell entity $r = 1111$ chooses A $x = (-r^2 \bmod n) = 2437$

. Then $T_e = 1116$ and computes $s = \{1,4,5,6\}$ chooses B Assume that entity

. Since: $y = 1282$ the number B and tells entity T $T_d = 96$ computes A entity

$$y^2 * T_e = 1282^2 * 1116 = 2437 = x \bmod n$$

, $r = 1990$ The verification condition holds. Similarity, the choices

. The $T_e = 688, T_d = 1228, y = 707$ give the values $s = \{2,3,5\}$, and $x = r^2 \bmod n = 256$

is satisfied. $-\, y^2 * T_e \equiv -2517 \equiv x \bmod n$ verification condition

3.2.4.6 Security of identification Protocol

The following security items can be discussed:

1. Probability of forgery: the proposed protocol is provable secure against chosen message attack in the following sense: providing that factoring n is difficult, the best attack has a probability 2^{-kt} of successful impersonation.
2. Security assumption required: The security relies on the difficulty of extracting square roots mod large composite integers n of unknown factoring. This is equivalent to that of factoring n.
3. Zero knowledge and soundness: The protocol is relative to a trusted server a sound zero knowledge proof of knowledge provided $k = O(\log\log n)$ and $t = O(\log n)$ regarding the practical significance of such constraints. A simplistic view for fixed k is that the verifier, interested in soundness, favors larger t more iterations for a decreased probability of fraud, while the prover interested in zero knowledge favors smaller t.
4. Parameter selection: choosing k and t such that $kt = 20$ allow a 1 in million chance of impersonation, which suffices in the case that an identification attempt requires a personal appearance by a would be impersonator. Computation memory and communication can be traded off, $1 \le k \le 18$ was originally suggested as appropriate. Specific

parameter choices might be for security $2^{-20}: k = 5, t = 4$: for $2^{-30} : k = 6, t = 5$.

5. Security trade-off: both computation and communication may be reduced by trading off security parameters to yield a single iteration $t = 1$ holding the product kt constant and increasing k while decreasing t . However, in this case the protocol is no longer a zero knowledge proof of knowledge.

3.3 Code definitions

In the section the author will describe the overall structure of the entire scheme. Figure 3.1 show the entire flowchart of the proposed schemes.

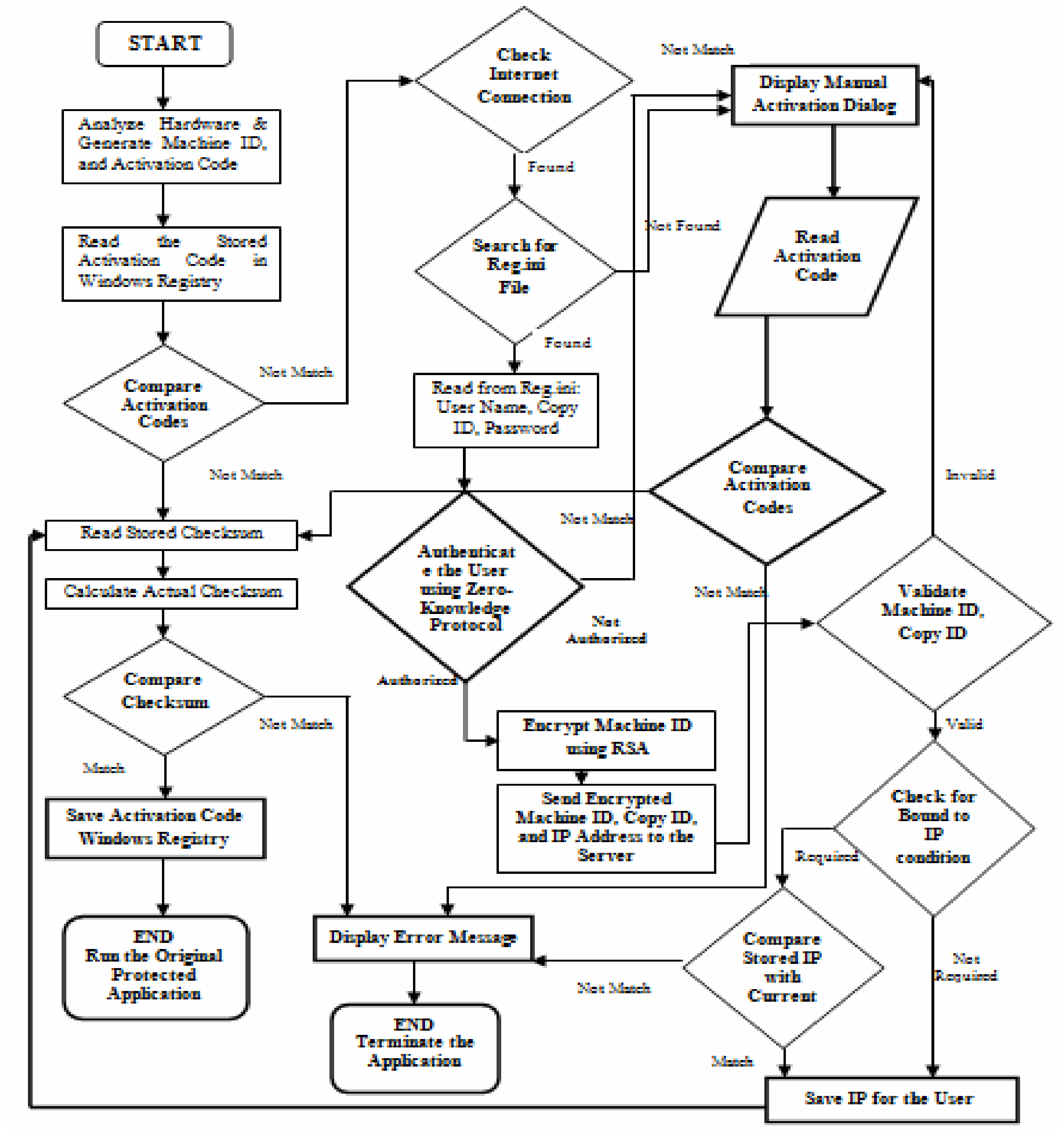
START
Analyze Hardware & Generate Machine ID, and Activation Code
Read the Stored Activation Code in Windows Registry
Compare Activation Codes
Not Match
Not Match
Read Stored Checksum
Calculate Actual Checksum
Compare Checksum
Not Match
Match
Save Activation Code Windows Registry
END Run the Original Protected Application
Check Internet Connection
Not Match
Found
Search for Reg.ini File
Not Found
Found
Read from Reg.ini: User Name, Copy ID, Password
Authenticate the User using Zero-Knowledge Protocol
Not Match
Not Authorized
Authorized
Encrypt Machine ID using RSA
Send Encrypted Machine ID, Copy ID, and IP Address to the Server
Display Error Message
END Terminate the Application
Display Manual Activation Dialog
Read Activation Code
Compare Activation Codes
Not Match
Invalid
Validate Machine ID, Copy ID
Valid
Check for Bound to IP condition
Required
Not Required
Compare Stored IP with Current
Not Match
Match
Save IP for the User

Figure 3.1 shows the entire flowchart of the proposed scheme.

3.3.1 Machine ID

The proposed scheme should read hardware serial numbers for ID , keyboard and hard disc, then generate a unique Machine CPU , $BIOS$ the

, keyboard and hard disc. Each serial number is $CPU, BIOS$ function to separated by a hyphen symbol. For example

= **xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx** ID Machine

The description of these 32 characters which are represented the computer address is as follows:

serial number. CPU The first 8 characters are the

serial number. $BIOS$ The second 8 characters are the

The fourth 8 characters are the keyboard serial number

The third 8 characters are the hard disc serial number

ID 3.3.2 Installation

will encrypt using the ElGamal public key ID The obtained machine is the message block that is the machine m encryption scheme, where . k_4 and k_1, k_2, k_3 and to be encrypted with ElGamal keys ID

ID 3.3.3 Copy

Each client could have unlimited number of copies for each purchased application. For example, ten copies from Microsoft Office. To identify that, ". This code is unique among items. So ID we added a code called "Copy is ID. The copy ID each software program could have its own and only copy mandatory for the activation process. It can generate such code using many ways. It is generated through a special developed algorithm. The ISCN is an identification number assigned to virtually every produced copy. It serves and links each one *IDs* to uniquely identify the copy. For a valid list of copy with a specific product sees table 3.1.

Table 3.1 Valid list of Copy IDs structure

Name Field	Field Type	Field Data Type	Description
ID	Primary Key	Number Auto	for each *ID* A unique Since it is a *ID* copy primary key, the value cannot duplicate in this table. This value is used for database indexing purposes.
ID Copy	Primary Key	Text	The actual copy value, since it is a *ID* primary key, the value cannot duplicated in this

ID Product	Foreign Key	Number	table of the *ID* The unique product which this copy uses, since that each *ID* product could use . *IDs* different copy Products are listed in table 3.2.

3.3.3.1 Products

The scheme can be used with several products. For that purpose table 3.2 is created. However, due to the nature of the system security more effort is done on security issues rather than on the sample database prototype.

Table 3.2 Products Structure

Name Field	Field Type	Field Data Type	Description
ID	Primary Key	Number Auto	for each *ID* A unique product. Since it is a primary key, the value cannot duplicate in this table.
Product Name	Primary Key	Text	The name of the product
Product			Short description for each product

Description	Primary Key	Text	

conditions *ID* **3.3.3.2 Copy**

are *ID* All copy IDs will store in company database. The conditions of copy as follows:

- Copy *ID* does not exist in company database => error message "invalid copy *ID*".
- Copy *ID* in company database, and
  1. No one used it before => continue with activation process that means add machine Information to the database.
  2. Used by the same user who is trying to activate the machine => continue with activation. But if that user tried to install the same copy and run the program from other machine so this means different hardware and thus different machine *ID* and Different Installation *ID*. The program will not run and error message will appear,"you cannot use the program on another machine".
  3. Used by another person on another machine => error message: "this copy is already registered"

3.3.4 Internet Protocol address

address is a unique address that certain electronic (*IP*) Internet Protocol devices use in order to identify and communicate with each other on a standard, in simpler terms, a computer *IP* computer network utilizing the address.

Address *IP* The proposed scheme has the ability to authenticate along with and allow or deny access to the company web server based on the public address is not always a unique *IP* of the client. In current practice, an *IP* identifier that always uniquely identifies a particular device, due to technologies such as dynamic assignment and network address translation.

address every time it connects to the *IP* When a computer uses the same addresses are manually *IP* address. Static *IP* network, it is known as a static assigned to a computer by an administrator. In contrast, in situations when address changes frequently such as when a user logs on to *IP* the computer a network through dialup or through shared residential cable it is called a address in dynamic mode is work in the same *IP* address. The *IP* dynamic way of the ISCN is work.

Address as an option and up to the *IP* The proposed scheme will keep using authentication such as schools, *IP* clients who are good candidates for libraries and other organizations that do not go through a common or Address and thus they will have more secure access to activate *IP* dynamic and run their owned copy and make it much harder to piracy.

3.3.5 Activation Code

This is the required code to run the system. For each client who has a there is a unique activation code will be in the company *ID* unique machine database with the end of transactions between the two parties and will be send back to the client only and only if the authentication processing is valid using zero knowledge proof of identity scheme. Also this code can be regenerated inside the proposed program. As it needs to regenerate this code inside the computer client in order to make the main comparison with what we have from the window registry. Table 3.3 shows the information those stores for each activation process of each customer. Each customer could have more than one activation process that is more than one program which is active using the proposed scheme, table 2.4 illustrated this property.

To generate a unique activation code, it needs the following steps:

- Read hardware serial numbers for the $CPU, BIOS$, keyboard and hard disc.
- Generate machine ID
- Encrypt step 2 using ElGamal scheme

Note: in step 2 we will use different keys (not the same keys that have been encryption), the ID from the machine ID used to generate Installation reason is, if it used the same keys anyone who can see the Installation and know the generation sequence for the activation code. ID

Table 3.3 Activations Structure

Field	Field	Field Data	Description

Name	Type	Type	
ID	Primary Key	Auto Number	for each activation *ID* A unique process. Since it is a primary key, the value cannot duplicate in this table.
CPU		Text	Store the serial number for .*CPU* them
Hard Disc		Text	Store the serial number for the hard disc
BIOS		Text	Store the serial number for .*BIOS* them
Keyboard		Text	Store the serial number for the keyboard.
Installation *ID*		Text	is stored directly *ID* The installation rather than computed to enhance the database performance. Although can be *ID* each installation generated from 4 serial. This will create a system over load when the user tries to fetch his list of activations.
Activation Code		Text	Store the same purpose as the .*ID* installation
IP Bond to		Boolean	This value tells the system whether to strict the program access to the given IP or not. It takes one of two values, true or false. This values will not be used if the *IP* user do not select bound to

			option
IP	Foreign Key	Text	of the customer *ID* This value stores who has this activation. Named foreign key to indicate that it is unique to other table
Customer *ID*	Foreign Key	Number	This value will connect the two tables that are user code and this table. Named foreign key to indicate that it is unique in the other table.
ID Copy		Number	

Table 3.4 Customer Structure

Field	Field	Field	Description

Name	Type	Data Type	
ID	Primary Key	Auto Number	for each customer. Since it is a primary ID A unique key, the value cannot duplicate in this table.
Name		Text	Customer name.
E-mail		Text	Customer email. The validation is held in the interface level (customer registration through the system webpage). So the interface (web page) will not insert any email address if it does not satisfy the naming rules for the emails (xxx@xxx.com).
Phone number		Text	Customer phone number. The validation will check if the number to the following international standard form xxx xxx xxxxxxxx not much the system will hold at the interface level.
Country		Text	Customer country. Rather than using the country name as a link value to the other table. It is used directly to indicate that the focus is not on the database design rather than on the overall system of security.
City		Text	Same as country.
Address		Text	The address of each customer.
User name		Text	The login name for the customer. No duplication allowed at the interface level.
Password	Primary	Text	The password for the customer. In the proposed scheme we do not store the direct password. Instead values that represent the e_A we store a series of public key for the customer. Where the direct values that represent the d_A password is a series of private key and is only known by the customers themselves. With this scheme we use zero knowledge proof of identity algorithm to authenticate the customer and there is no way for any intruder on the database to steal the passwords of customers. This

	key		show the high security level of this scheme.

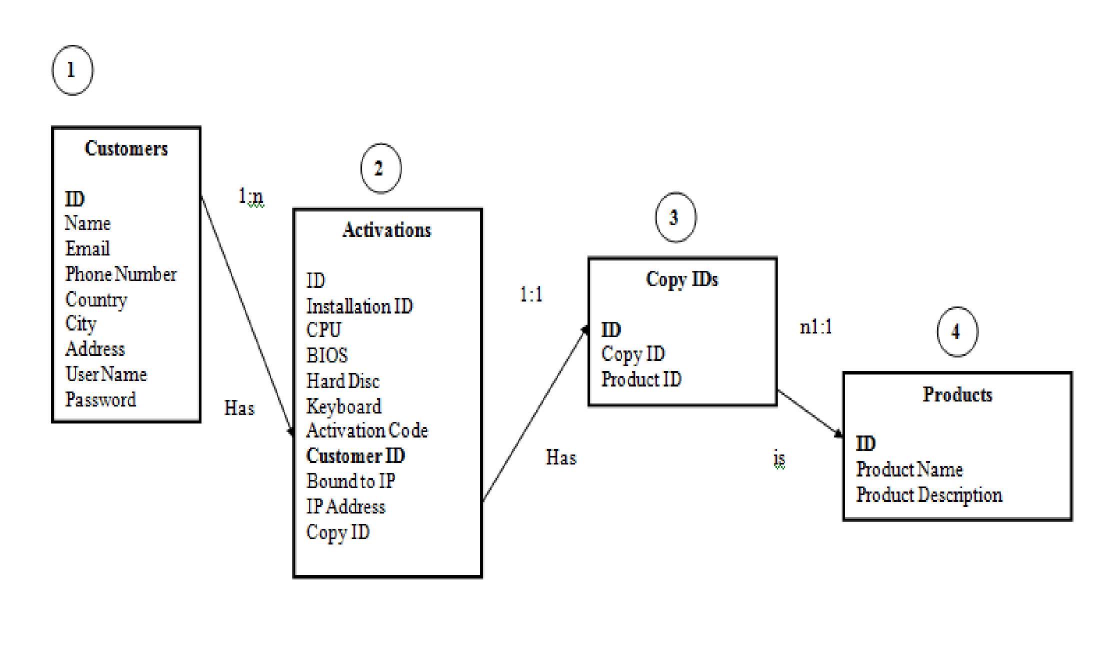

Figure 3.2 shows the code Definition

3.4 Obtain the Hardware Information

- The company system already obtained the hardware information for the client computer because the computer has been sent to the company. The company will obtain the hardware information that is the serial numbers for the $CPU, BIOS$, keyboard and hard disc, install the software on client computer and send it back to the client ready to use. Also computer assembly manufactures can bundle the application with other software packages such as Windows, Microsoft Office as preinstall software.
- If it could not send before the client computer to the company to install the software, the client should install the software manually. After the installation the client will access the Internet and thus the company system will obtain the hardware information at the first run of the program, bases on that the copy will be locked for a specific user on his specific machine.

If there is no Internet connection, the client can update the company with the hardware information by telephone or fax as he will inform them the which the client will have through the ID and the Copy ID installation program execution on his computer.

3.5 Authentication

Activation process takes place by generate and send an Activation code in case the client has:

- Validate Copy ID (not already used and an error message will appear if the system dictate that)
- Validate Installation ID (Once the company obtained Machine ID, it will be checked with the company Data Base to detect wither the client is installing the software package on his machine or on another machine).
- Validate IP Address (our application will consider this validation if and only if the client activate IP Address authentication).
- Validate File Checksum value (any tampering in the file will lead to failure in the authentication process)

Any fail in steps above, company will know that the user is using a pirated copy. No activation code will send in this case.

3.6 Customer Tracking System

In this section the author will describe the tracing system which is as follows:

3.6.1 Database

Store the information about the clients and their purchased products. This database can be accessed through either the web service or through the web interface. Here, database created specifically to enable us testing the proposed scheme. The database and its system could be extended in several ways. However, this is out of the proposed scheme scope.

Entity relationship model

This is an abstract conceptual representation of structured data. Entity relationship modeling is a relational schema database modeling method, used to produce a type of semantic data model of a system, often a relational database, and its requirements in a top down fashion, see figure 3.4. Table 3.5 shows the relations defined in the proposed entity relationship model.

Table 3.5 Entity Relationship Model

Relation	Type	Description

Customer has an Activations	$1:n$	Each customer could have one or more associated activations
ID Activation has a copy	$1:1$	Each activation process uses only one associated *ID* copy
is a product *ID* Copy	$n:1$	could be *IDs* Many copies of one product

3.6.2 Website

1. **Register New User Page**: create a typical set of information record for each client.
2. **Login Page**: access to the database using the proposed protocol identification the zero knowledge proof of identity.
3. **Product Page**: list all the client activated product.
4. **New Copy Activation Page**: a sub page from the product page

3.6.3 Web service

The essential part of web service is the interact relationship between a service provider and service requestor. So when the author need to get some programming task done. It can make use of a web service by calling it over the Internet. However, by pass the parameter with the request. It can expect to receive a response containing the result generated by the web service.

Web sites are just the user interface of your application. Web service is intended to expose some functionality to the outside or to some other layer as a service. The author is using the web service in the path where the protection interface needs to interact with the server directly using the proposed secure protocol Zero knowledge proof of identity in order to activate the copy. The client is directly connected to the network and has a registration file in the directory. This is the case which the client has already sent the pc to the company.

3.7 The phases of the proposed scheme

The previous related studies either focused on creating a secure environment which demands lots of prerequisites from the customers. Such as being always connected to a network, using Linux environment, or using a specified secure execution policy on a certain windows system. Some other studies made secure execution methods. But, they were relying on costly hardware devices which lead to other software complications such as incompatibility with operating systems updates.

In this thesis, the author is about to implement a better scheme for software protection. This technique combines several security models to achieve the objectives. Rather than creating an environment that is not widely applicable the author is presenting an environment that can work everywhere. Figures 3.6 show the phases of the proposed scheme.

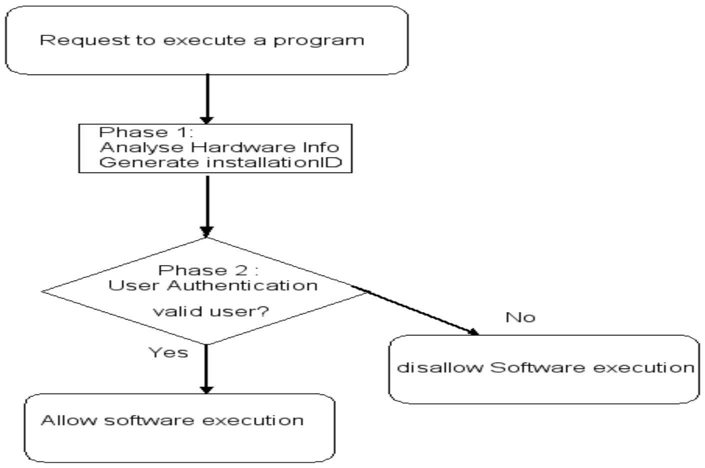

Figure 3.3 Illustrate the phases of the proposed scheme

At the first phase. The scheme begins with the customer computer where the protected software analysis the hardware of the client. Then it produces a code which combines several read hardware serials such keyboard, and hard disc. The code is then encrypted using $BIOS,, CPU$ as Elgamal public key encryption scheme. The resulted code will be named $.ID$, or Machine ID Installation

In the second phase the code will be send to the development company by either one from two methods:

1. Automatically via internet connection. Either by a registration page or directly without informing the user using a secure model.
2. Manually by phone.

Once the code is received at the development company, the code will be analyzed and then decrypted In order to re-obtain the hardware information of the customer. Now the obtained values will be checked using another provided system to detect without the user installing the system on his machine or on another machine. Now we have two scenarios:

1. Valid: The user is installing the software on the machine. Now the company will send a code called activation code to the customer. When the user uses this code the program will run on the machine.
2. Not Valid: If the user is using the software on a different machine then the values will not match and the development company will know that the user is using a pirated copy. No activation code will be send in this case.

Hardware Analyzer

This is the part which analysis and encrypts the hardware serials. It uses public key algorithm to encrypt the values. It is the main component of the installation: An implementation of public key $f(x, y, z, w) =$ proposed system: algorithm.

Customer tracking system

This is a small system which stores information about the users and their equivalent hardware serials. The system will be used to identify without the user is legal or not.

Notation used

- Installation ID : the encrypted hardware serials.
- Activation Code: the code that will enable the software to work on the user machine.
- x : Hard disc-serial number (16 chars long).
- y : CPU-serial number (16 chars long).
- z : Bios Serial Number (16 chars long)
- w : Keyboard serial number (16 chars long)

- $f(x, y, z, w)$: Encryption function which is an implementation of public key algorithm.
- User ID : unique user identification number for the manual mode.

Operation Modes

The system can work in two modes:

1. Automatic mode: In this mode messages is shown to the user. This mode is optimal for the case where software is already bundled in the pc like in laptops. The hardware information is already saved in the database system in this case.
2. Manual Mode: this mode displays the obtained codes and demands the user to submit them to the delivery company via internet registration. This mode is optimal in the case of software that is not pre-installed in the client machine. In this case an additional user ID will provide with each copy to allow the software registration. The hardware information will be obtain at the first run of the program and based on that the copy will be locked for a specific user on a specific machine.

Requirements

If you wish to program this scheme you have to provide a modern computer with 3.0 GHz CPU and 2 GB RAM utilizing tools on the Windows . $.net2.0$ operating system, It means the one that is capable of running the and the protected $VB.net$ The proposed program can be implemented using . XP application will work on win

Chapter Four: Tools Used

4.1 Introduction

the Z_n Many public key encryption schemes require computations in is a large positive integer which may or may not be a n (n integers mod prime). For example RSA and ElGamal schemes require efficient methods is Z_n. Although Z_n for performing multiplication and exponentiation in prominent in many aspects of modern applied cryptography, other algebraic structures are also important. These include but are not limited to, polynomial rings, finite fields and finite cyclic groups. The efficiency of a particular cryptography scheme based on any one of these algebraic structures will depend on a number of factors, such as parameter size, time memory tradeoffs, processing power available, software or hardware optimization and mathematical algorithms [40].

This chapter is concerned primarily with mathematical algorithms for efficiency carrying out computations in the underlying algebraic structure. , Z_n Since many of the most widely implemented techniques rely on emphasis is placed on efficient algorithms for performing the basic arithmetic operations in the structure. Efficiency can be measured in numerous ways. Thus, it is difficult to definitively state which algorithm is the best. An algorithm may be efficient in the time it takes to perform a certain algebraic operation, but quite inefficient in the amount of storage it requires. One algorithm may require more code space than another. Dependent on the environment in which computations are to be performed, one algorithm may be preferable over another. For example, current chip-card technology provides very limited storage for both pre-computed values and program code [39]. For such applications, an algorithm which is less efficient in time but very efficient in memory requirements may be preferred.

The algorithms described in this chapter are those which for the most part, have received considerable attention in the literature. Although some is made to point out their relative merits, no detailed comparisons are given.

Z_p^* and generator of p 4.2 Selecting a prime

is required, one Z_p^* In cryptography applications for which a generator of . To guard against the p usually has the flexibility of selecting the prime ElGamal algorithm for computing discrete logarithms, a security

. In this q should contain a large prime factor $p-1$ requirement is that represents an infeasible amount $\sqrt{q}$ context, large means that the quantity . This suggests the following algorithm $q \geq 2^{160}$ of computation, for example .(p,a) for selecting appropriate parameters

is q where $p = 2Rq+1$ is a prime of the form p **Definition** A safe prime prime, using algorithm 4.1.

Z_p^* of a and a generator p Algorithm 4.1 generates a safe probable prime

---

Z_p^* of a and a generator p -bit prime k 4.1 Algorithm: Selecting a

---

of the prime k Input: the required bit length

Z_p^* of a and a generator p -bit safe prime k Output: a

1. Repeat the following:

(for example using Algorithm 4.1.1) q bit prime $k-1$ 1.1 Select a random

is prime or not using trial p and test whether $p = 2q+1$ 1.2 Compute

Division by small primes and algorithm 4.1.2

is prime p Until

Z_p^* of a 2. Use Algorithm 4.1.3 to find a generator

(p,a) 3. Return (

To satisfy the step of 1.1 above it should describe the algorithm 4.44 that it used for generating probable primes.

4.1.1 Algorithm: Random search for a prime using the Miller-Rabin test

(k,t) RANDOM SEARCH

t and a security parameter k INPUT: an integer

-bit probable prime k OUTPUT: a random

1. Generate an odd k-bit integer n at random
2. Use trail division to determine whether n is divisible by any odd prime $\leq B$ if it is then go to step 1
3. If MILLER-RABIN(n,t) Algorithm 4.1.2 output prime then return(n). Otherwise go to step 2

4.3 Miller-Rabin Test

To satisfy the step 1.2 of the algorithm **4.1** and the step number 3 of the algorithm **4.1.1.** It should describe the Miller-Rabin algorithm since the entire probabilistic primality test used at most the Miller-Rabin test. The test also known as the strong pseudo prime test this test is described on the following Algorithm.

4.1.2 Algorithm: Miller-Rabin probabilistic primality test

(n,t) MILLER RABIN

t and a security parameter $n \geq 3$ INPUT: an odd integer an integer

prime. n OUTPUT: an answer prime or composite to the question is

```
1. Write $n-1=2^s * r$ such that $r$ is odd
2. For $i$ from 1 to $t$ do
   2.1 Choose a random integer $a, 2 \leq a \leq n-2$
   2.2 Compute $y = a^r \bmod n$ using algorithm 4.2
   2.3 If $y \neq 1$ and $y \neq n-1$ then do the following:
         $j = 1$
         While $j \leq s-1$ and $y \neq n-1$ do the following:
               Compute $y = y^2 \bmod n$
               If $y = 1$ then return (composite)
               $j = j+1$
         If $y \neq n-1$ then return (composite)
3. Return (prime)
```

4.3.1 Example

$t=1$ and security parameter $n=17$ Suppose

1. $16 = 2^4 * 1, \therefore r = 1$ odd
2.

$a = 2$ 2.1 Suppose

$y = 2^1 \bmod 17 = 2$ 2.2 Compute

$j = 1$

While $j \leq s-1$ and $y \neq n-1$ True
Compute $2^2 \bmod 17 = 4$
$j = 2$
While $j \leq s-1$ and $y \neq n-1$ True
Compute $4^2 \bmod 17 = 16$
$j = 3$

3. Return (prime)

4.4 Square and Multiply Algorithm

Modular exponentiation can be performed efficiently with the repeated square and multiply algorithm (Algorithm 2.143), which is crucial for many cryptography protocols. One version of this algorithm is based on the , $\sum_{i=0}^{t} k_i * 2^i$ be k following observation. Let the binary representation of . Then $k_i \in \{0,1\}$ where each

$$a^k = \prod_{i=0}^{t} a^{k_i * 2^i} = (a^{2^0})^{k_0} (a^{2^1})^{k_1} \ldots (a^{2^t})^{k_t}$$

4.2 Algorithm: Repeated Square and Multiply Algorithm for Z_n exponentiation in

whose binary representation is $0 \le k < n$ and integer $a \in Z_n$ INPUT:

$k = \sum_{i=0}^{t} k_i * 2^i$

$a^k \bmod n$ OUTPUT:

```
1. Set b=1. If $k = 0$ then return ($b$)
2. Set $A = a$
3. If $k_0 = 1$ then set $b = a$
4. For $i$ from 1 to $t$ do
    1. Set $A = A^2 \bmod n$
    2. If $k_i = 1$ then set $b = A * b \bmod n$
5. Return ($b$)
```

4.4.1 Example

$5^{596} \bmod 1234 = 1013$ Table 4.1 shows the steps involved in the computation of

9	8	7	6	5	4	3	2	1	0	i
1	0	0	1	0	1	0	1	0	0	k_i
925	947	779	421	369	1011	681	625	25	5	A
1013	1059	1059	1059	67	67	625	625	1	1	b

Z_p^* of a 4.5 Algorithm 4.1.3 to find a generator

Before it is going to discuss this algorithm it should study the generator of a cyclic group.

. Then for any divisor of n is a cyclic group of order G Suppose now that is the θ, where $\theta(d)$ is exactly G in d the number of elements of order n generators and hence $\theta(n)$ has exactly G Euler phi function. In particular, . Using $\theta(n)/n$ being a generator is G the probability of a random element in the low bound for the Euler phi function, this probability can be seen to be . This suggests the following efficient randomized $1/(6\ln\ln n)$ at least algorithm for finding a generator of a cyclic group.

4.1.3 Algorithm: Finding a generator of a cyclic group

$n = p_1^{e_1} * p_2^{e_2} \ldots p_k^{e_k}$ and the prime factoring n of order G INPUT: a cyclic group

G of a OUTPUT: a generator

```
1. Choose a random element $a$ in $G$
2. For $i$ from 1 to $k$ do
   Compute $b = a^{n/p_i}$
   If $b = 1$ then go to step 1
3. Return ($a$)
```

4.5.1 Example

5 and $n = 2$, so the factors of $p = 11, \therefore \theta(n) = 10$ Suppose that

1. Suppose that $a = 2$
2. Compute $b = 2^{10/2} \bmod 11 = 10$, compute $b = 2^{10/5} \bmod 11 = 4$
3. $\therefore a$ is a generator

(gcd) 4.6 Greater common Divisor

can be computed b and a The greater common divisor of two integers efficiently by Euclidean. The Euclidean 4.3 is an efficient algorithm for computing the greater common divisor of two integers that does not require the factorization of the integers. It is based on the following simple $\gcd(a,b) = \gcd(b, a \bmod b)$, then $a \geq b$ are positive integers with b and a fact. If

4.3 Algorithm: Euclidean Algorithm for Computing the Greater common (gcd) Divisor

$a \geq b$ with b and a INPUT: two non-negative integers

b and a OUTPUT: the greater common divisor of

```
1. While $b \neq 0$ do
```

```
    1.1 $r = a \bmod b$
    1.2 $a = b$
    1.3 $b = r$
2. Return $(a)$
```

bit operation. $O((\log n)^2)$ The above algorithm has a running time of

4.6.1 Example

The following are the division steps of algorithm 4.3 for computing $\gcd(4864,3458) = 38$

$$4861 = 1*3458 + 1406$$

$$3458 = 2*1406 + 646$$

$$1406 = 2*616 + 114$$

$$646 = 5*114 + 76$$

$$114 = 1*76 + 38$$

$$76 = 2*38 + 0$$

4.7 Inverse

The Euclidean algorithm can be extended to calculate the inverse of d (algorithm 4.4) so that it not only yields the greater common divisor . $ax + by = d$ satisfying y and x, but also integers b and a two integers

4.4 Algorithm: Euclidean Algorithm for Computing the Inverse

$a \ge b$ with b and a INPUT: two non-negative integers

$ax + by = d$ satisfying y and x and integers $d = \gcd(a,b)$ OUTPUT:

```
1. If $b = 0$ then
        $d = a, x = 1, y = 0$
        Return $(d, x, y)$
2. Set $x_2 = 1, x_1 = 0, y_2 == 0, y_1 = 1$
3. While $b > 0$ do
    3.1 $q = \lfloor a/b \rfloor, r = a - q * b, x = x_2 - q * x_1, y = y_2 - q * y_1$
    3.2 $a = b, b = r, x_2 = x_1, x_1 = x, y_2 = y_1, y_1 = y$
4. $d = a, x = x_2, y = y_2$, Return $(d, x, y)$
```

4.7.1 Example

. $b = 3458$ and $a = 4864$ Table 4.2 shows the steps of algorithm 4.4 with input

$(4864)(32) + (3458)(-45) - 38$ and $\gcd(4864, 3458) = 38$ Hence

y_1	y_2	x_1	x_2	b	a	y	x	r	q
-	-	-	-	4864	3458	1	0	0	1
-1	1	1	0	1406	3458	-1	1	1406	1
3	-1	-2	1	646	1406	3	-2	646	2
-7	3	5	-2	114	646	-7	5	114	2
38	-7	-27	5	76	114	38	-27	76	5
-45	38	32	-27	38	76	-45	32	38	1

Chapter Five: Conclusion and Future Work

5.1 Conclusion

In this thesis the author presented a new scheme for the protection of software against piracy. Its strength is based on combined serial numbers of some computer components with ElGamal public key encryption scheme, each installed copy is unique. Updates are tailored to fit one instance and one instance only. This way the ease with which a useful additional copy can be created is diminished. Furthermore, as illegitimate instances cannot be kept sound and up to date unless a new line of defense is broken with every critical update, this results in a dynamic nature of defense. It can point out the improvements over previous approaches and argued that it makes most forms of software piracy more difficult in a realistic model under realistic assumptions.

In this thesis along with its developed applications, the author managed to create a piracy prevention technique that will help the developer, end user and high security enterprise users. The developer can now use the proposed scheme to protect the future products with a very easy way. Also, the end user can implement the protected scheme with no obstacles. The enterprise user can experience new levels of security. The proposed scheme generated as a structure for future schemes where other people can build and extend its features in the same context. The author combined well known techniques in addition to implementation other proposed techniques especially the identification protocol which hope to be used in other schemes as well.

5.2 Future Work

In this section, the author describes some of the future enhancements that the proposed scheme could have:

Used as a Customizable Framework: Any part of the scheme could be enhanced in one way or another in the future. Since the proposed scheme was designed to address all parts of the equation. Each part (developer, end user, enterprise user) can have an additional feature that suits him better. This would be implemented either using another research or inside development company.

Licensing Management: The framework could benefit from another *ID* competing idea such as advanced license management where the copy bound to it. *IP* could allow more than one license, or more than one Another idea is the expiration date for each license. It can manage to bind some products to an expiration date or license renewal period.

Virus Attack Rather than just an Error Message: In the current implementation the author used to display an error message in the case of failures. But it can prevent piracy with another idea in the future. A virus like program would be executed in the scheme rather than just informing the user that he is an unauthorized user. It might have some sort of attack on the scheme either a friendly attack such as displaying an annoying pop up each couple of seconds or more seriously to destroying some important files in the system. Such idea could be used as a defense mechanism for high security situations where only the authorized user is allowed to use such important program and in any other cases the scheme will self destruct.

Customer Relationship Management System: Real life situations will not make greater usage of the embedded customer tracking system due to its prototype nature. In such situations it could build a complete customer relationship management solution where more detailed information are stored for each user and the ability to sell products that benefit from the suggested techniques online. An additional automated customer support system is a welcomed idea. Many well known small features can be added to such scheme.

References

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

[2] Altinkemer K., Guan J., "Analyzing protection strategies for online software distribution", Journal of Electronic Commerce Research, 4(1):34–48, 2003.

[3] Ashileshwari Chandra, Liam Comerford, Steve White, "Software Protection System Using Single-Key Cryptosystem", A Hard-based, IBM, 1989.

[4] 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.

[5] Bertrand Anckaert, Bjorn De Sutter, Koen De Bosschere, "Software Piracy Prevention through Diversity", Proceedings of the 4th ACM workshop on Digital Rights Management, Washington DC, USA, November 2004, pp. 245-256.

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

[7] 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, 2002.

[8] Clyde W. Holsapple; Deepak Iyengar; Haihao Jin; Shashank Rao, The "Parameters for Software Piracy Research", Journal of July 2008, pp. 199 – 218, 4 , Issue 24, Volume Information Society USA.

[9] Collberg C. and Thomborson C., "Software watermarking: Models and dynamic embeddings", In Principles of Programming Languages, pages 311–324, 1999.

[10] Cowan C, “Software Security for Open-Source Systems", IEEE Security and Piracy, 2003.

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

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

[13] Digital millennium copyright act of 1998. Publication, No. 105-304, 112 Stat. 2860, October 28, 1998.

[14] Diffie W., and Hellman M, "New Direction in Cryptography", IEEE Transaction on Information Theory, 1976.

[15] Eldad Eilam, "Reversing: Secrets of Reverse Engineering", pp. 312-314, John Wiley & Sons, 2005.

[16] Electronic Frontier Foundation, Unintended consequences: Five years under the dmca, September 24, 2003. www.eff.org.

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

[18] Eric Kin, wai Lau, "Interaction effects in software piracy", Business Ethics: A European Review, Volume 16, No. 3, pp. 34-47, **2007.**

[19] Felten E., "Understanding trusted computing: will its benefits outweigh its drawbacks", IEEE Security and Privacy, 1(03):60–62, 2003.

[20] Govindavajhala S. and A. Appel, "Using memory errors to attack a virtual machine", In IEEE Symposium on Security and Privacy, 2003.

[21] Horne B., Matheson L., Sheehan C., Tarjan R., "Dynamic self-checking techniques for improved tamper resistance", Security and Privacy in Digital Rights Management, LNCS, 2320:141–159, 2002.

[22] IDC, "Reducing Software Piracy Could Have Exponential Effect on Channel Profitability", HOUSTON — July 9, 2008.

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

[24] Jakobsson M., Reiter M., "Discouraging software piracy using software aging", Security and Privacy in Digital Rights Management, LNCS, 2320:1–12, 2002.

[25] Jon H. Barber, Ronald A. Woodward, Richard M. Buekley, Erwin L. Rehme, Mathew W. Jackson, Douglas M. Young, “System for Controlling the Number of Concurrent Copies of a Program in a Network Based on the Number of Available Licenses”, United States Patent 5390297, 1995.

[26] Kizza, Joseph Migga, "Ethical and Social Issues in the information Age", 3rd Edition, Springer Verlag, 2007.

[27] Lie D., Thekkath C., Mitchell M., Lincoln P., Bohen D., Mitchell J., Horowitz M., "Architectural support for copy and tamper resistant software", In Architectural Support for Programming Languages and Operating Systems, pages 168-177, 2000.

[28] Mathew Schmid, Frank Hill and A. Ghosh, “Preventing the Execution of Unauthorized Win32 Applications", DARPA International Survivability Conference and Exposition (DISCEX II’01) Volume II, p. 1175, 2000.

[29] Michael Stolpe, "Protection Against Software Piracy: A Study Of Technology Adoption For The Enforcement Of Intellectual Property Rights", Economics of Innovation and New Technology, Volume 9, Issue 1, 2007 , pp. 25 – 52, USA.

[30] Nagra J., Thomborson C., Collberg C., "A functional taxonomy for software watermarking", In M.J. Oudshoorn, editor, Twenty-Fifth, Australasian Computer Science Conference (ACSC2002), ACS, 2002.

[31] 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. 21-32, 2006.

[32] Oorschot P. van, "Revisiting software protection. Information Security", LNCS, 2851:1–13, 2003.

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

[34] Rivest R, Shimie A, and Adleman L, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, 1978.

[35] Salomon D., "Coding for Data and Computer Communications", Springer, 2005

[36] Schaefer E., "A Simplified Data Encryption Standard Algorithm", Crypto-logia, 1996.

[37] , Catherine D. Marcum, "Deterrence George E. Higgins, Scott E. Wolfe and Digital Piracy", Social Science Computer, Volume 26, Issue 3 (August 2008), Pages 317-333, Sage Publications, Inc. USA.

[38] Software piracy protection device, Document Type and Number: United States Patent Application 20060185020, 2006, USA.

[39] Stalling W., "Cryptography and Network Security", 3rd Edition, Prentice-hall, 2006

[40] Stinson D., "Cryptography Theory and Practice", 3rd Edition, CRC, 2006

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

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

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

[44] 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.

[45] van Oorschot, "Revisitin software protection", Information Security, *LNCS*, 2851:1–13, 2003.

[46] "IPR Protection for Digital Media Vassiliadis B, Fotopoulos V, Distribution: Trends and Solutions in the E-Business Domain", International Journal of E-Business Research, Volume 3, Issue 4, 2007, pp. 79-97, USA.

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

[48] Zoeller, Renate, "Nest of Pirates", Transactions Online: 8/27/2007, p 5.

Appendix A

ElGamal Public Key Encryption Scheme

The ElGamal scheme like the RSA scheme where both are the most employed public encryption compared with the other schemes. Also, both can be employed for both encryption and digital signature schemes. So in the proposed scheme we use the ElGamal scheme. The description of ElGamal scheme is as follows:

1. Algorithm for Key generation

must do the following: A To generate the keys entity

1. Generate a large random prime number p by which we mean one with equally around 1024 bits, such that $p-1$ is divisible by another medium private q of around 160 bits and a random integer generator a an element of the multiplicative group Z_p^* of the integer mode p (using algorithm 4.1 in chapter 4)
2. Select a random integer $x, 1 \le x \le p-2$, which is representing the private key
3. Compute the public key $h = a^x \bmod p$ (using algorithm 4.2 in chapter 4).
4. Determine entity A's public is (p, g, h); and A's private key is x.

2. Algorithm for Public key encryption

This algorithm involved two algorithms which are as follows:

should do the following: B - **Encryption Algorithm:** entity

1. Obtain entity A's public key (p, g, h)
2. Represent the message m as an integer in the interval $\{0,1,\ldots, p-1\}$
3. Select a random integer $k, 1 \le k \le p-2$
4. Compute $y = a^k \bmod p$
5. Compute $g = m * h \bmod p$
6. Send the cipher-text $c = (y, g)$ to entity A.

- Decryption Algorithm

should do the following A, entity c from m To recover the message

1. Use the private key x to compute $y^{p-1-x} \bmod p$
2. Recover m by computing $y * g \bmod p$

Proof: To prove the decryption works. The decryption of algorithm described above allows recovery of original message since:

$$y^{-x} * g \equiv a^{-x*k} m * a^{x*k} \equiv m \bmod p$$

3. Example

selects the prime numbers p = 43 and a A **Key generation:** Suppose entity and $x = 8$ chooses the private key A. Then entity Z_{43}^{*} of $a = 2$ generator $(p = 43, a = 2, h = 41)$ public key is A entity $h = a^{x} \bmod p = 2^{8} \bmod 43 = 41$ compute . $x = 8$'s private key is A while

and $(p = 43, a = 2, h = 41)$ obtain A's public key B**Encryption**: Suppose entity selects a random integer B, entity $m = 12$ he determines a message B. Entity $g = 12 * 41^{19} \bmod 43 = 3$ and $y = 2^{19} \bmod 43 = 32$ and computes $k = 19$ A to entity $g = 3)$ and $c = (y = 32$ then send

Decryption: To recover and obtain the original message m entity A should the following:

then computes $B$from entity $g = 3)$ and $c = (y = 32$ Obtain the cipher text recover the original A and then entity $y^{p-1-x} = 32^{34} \bmod 43 = 4$ the following . $m = 4 * 3 \bmod 43 = 12$ as follows m message

4. Analysis of ElGamal Scheme

The user needs only one prime to generate a random number and perform a modular exponentiation, while in RSA for example each user needed to generate a key pair two large primes to set up their key pair which is a costly task. This means that the ElGamal scheme is more efficient and more secure. This will be discussed in the following section.

5. Efficiency of ElGamal Scheme

The encryption process requires two modular exponentiations, namely . These exponentiations can be speeded up by $(a^x)^k \bmod p$ and $a^k \bmod p$ having some additional structure, for k selecting random exponents example having low Hamming weights. A disadvantage of ElGamal encryption is that there is message expansion by a factor of 2. That is, the cipher text is twice as long as the corresponding message.

in which a and generator p All entities may elect to use the same prime need not be published as part of the public key. This results in a and p case public keys of smaller sizes. An additional advantage of having a fixed base is that exponentiation can then be expedited by pre-computations using a the techniques called fixed base exponentiation algorithm (windowing method, fixed base Euclidean method, or fixed base com method). A potential disadvantage of common system wide parameters is that larger may be warranted. p mod

a 512 bit Z_p^* Given the latest progress on the discrete logarithm problem in provides only marginal security from concerted attack. As of a p modulus of at least 768 bits are recommended. For long term security, p modulus 1024 bit or larger mod should be used. For common system wide parameters even larger key sizes may be warranted. This is because the document stage in the index calculus algorithm for discrete logarithm in is the pre-computation of a database of factor base logarithms, Z_p^* following which individual logarithms can be computed relatively quickly. Thus computing the database of logarithms for one particular modulus . p will compromise the secrecy of all private keys derived using p

6. Security of ElGamal Scheme

The problem of breaking the Elgamal encryption scheme that is recovering is equivalent to solving the discrete (p, a, h, y, g) given m the message logarithm problem. For this reason, the security of the ElGamal encryption . Z_p^* scheme is said to be based on the discrete logarithm problem in

be used to encrypt different k It is critical that different random integers and m_1 is used to encrypt two messages k messages. Suppose the same . (y_2, g_2), and (y_1, g_1) and the resulting cipher-text pairs are m_2 were known. m_1 could be easily computed if m_2, and $g_1 / g_2 = m_1 / m_2$ Then

ElGamal encryption is one of many encryption schemes which utilize randomization in the encryption process like any other probabilistic encryption schemes. Deterministic encryption schemes such as RSA may also employ randomization in order to circumvent some attacks. The functional idea behind randomized encryption techniques is to use randomization to increase the cryptography security of an encryption process through one or more of the following methods:

1. Increasing the effective size of the message space

2. Precluding or decreasing the effectiveness of chosen-plaintext attacks by virtue of a one-to-one mapping of message to cipher text