**COMBATING SOFTWARE PIRACY USING CODE ENCRYPTION TECHNIQUE**

**By**

**AgamalafiyaBulus Jonathan**

**PGS/02/17202604**

**(CHAPTER 1 to 3)**

**Main Supervisor: Dr. Garba Muhammad**

**Co-Supervisor: Dr Mairo Danjuma**

**Co-Supervisor: Mal. Anas Mohammed Gulumbe**

**Department of Computer Science**

**Kebbi State University of Science and Technology, Aliero**

# CHAPTER ONE

**INTRODUCTION**

### Background to the Study

Software piracy has affected the growth and profitability of the software industry. It causes so much harm to software publishers when their products are duplicated without due authorization and spread across to individuals who are not licensed users or when their products are installed amongst multiple machines beyond the licensed number. Nowadays, we are surrounded by software applications which are used for services like online payments, social networking, games, data management, etc. The profitability of the producers and vendors of these software applications are negatively affected by threats such as piracy, reverse engineering, and tampering.

However, these threats are exacerbated when these software are poorly protected. Therefore, it is important to have a thorough threat analysis as well as software protection schemes in order to protect the huge revenues of software companies which increases in proportion with the wide use of software. This protection is not only for operating systems, but also for professional applications (e.g. graphics software) which can also be very expensive. The illegal use of software is on the increase and with just a few mouse clicks, people can download software product, apply a downloaded patch to it, and start using it without payment. Vendors realized that protecting software against malicious users is a hard problem. The user is in control of his machine: he has physical access to the hardware and he controls the network connectivity (Howard & LeBlanc, 2002).

The BSA, (2013). explained that software piracy creates unfair competition for legitimate companies, damages brands through distribution of substandard products, and exposes customers to a range of IT risks including malware, security breaches and data loss. It also denies end-users the benefit of necessary customer support, upgrades, technical documentation, trainings and bug fixes. These pirated software end-users of course have no warranty to protect themselves and they are subject to significant fines for copyright infringement. The software vendors and manufacturers and also the BSA with local laws in different countries have not relented in the quest to completely prevent software piracy. Many websites today are used to fight piracy by including linking information such as “report software piracy”, thereby involving legitimate users of software in the fight. Software piracy is more often practiced when digital files are made into one or more copies, thus programmers have usually come together trying to find a way of making digital files uncopyable. In his work, Schneier, (2013). Has argued that trying to make digital files uncopyable is like trying to make water not wet.

(Moores and Esichaikul, 2011). has also argued that there is no good technological solution to a behavioural problem. While so many are still on the fight for piracy prevention, some others are arguing that software piracy prevention has no perfect measure. According to Warner and Sloan, (2012). A combination of four main factors means that no imperfect copy-protection system can possibly be safe. In the light of his “four major factors”, he said the question shouldn’t be “Why are software still easily pirated?”, but “Why are people still trying to prevent it?”. So many others do believe that piracy can be tackled with a combination of measures. A decision to totally relent in the fight to prevent software piracy is like a decision to cease software production all over the world and its impact will be unimaginable.

### 1.2 Statement of the Problem

The major Problems of arising from software piracy that triggered this research are as follows:

**Loss of jobs:** Unfortunately today, throughout the world, we do not only have a problem of software piracy reducing this funding of ongoing development efforts, but also a problem that software vendors have not really achieved the desired much in their quest to prevent this dangerous menace which affects their profitability and which also is a threat to the jobs of so many software developers.

**Financial losses to software firms and the government:** The BSA study concluded that software piracy cost the U.S economy $1.9 billion and resulted in more than 105,000 lost jobs (Bishop, 2004). At global level, the global piracy rate hovered at 42% in 2011 while a steadily expanding marketplace in the developing world drove the commercial value of software theft to $63.4 billion (BSA, 2011). Anderson, *et al,* (2013). also stated that more than $11 billion is lost to piracy every year.

**Investment discouragement:** Software piracy has also been a discouragement to so many who wishes to invest in the software industry thus affecting the expected alternative software solutions to solve people’s needs in the future. If enough strict software piracy–prevention measures are not put in place, it could lead to a very high increase in software prices in the future because of the need for business sustainability on the side of software vendors.

### Aim and Objectives of the Study

This study is aimed towards achieving the following results:

1. To investigate the state of the art research in combating software piracy using code encryption techniques
2. To investigate the key challenges involved in combating software piracy
3. To critically analyze using Chi-Square the significant of the encrypted software over the non-encrypted software.
4. To investigate the financial losses due to software piracy activities.
5. To proffer possible solution to the above problems.

### Research Questions

Would it be that software users are not educated about software piracy or does it mean it still appeals to them ethically to still engage in software piracy even when they are educated enough about software piracy and its effect on the world economy. Would it not be appropriate for software vendors to work very hard in order to safeguard their products even if it affects the time they should have put in the development of new products or enhancing available ones for a better users’ experience? Most software products initially come with some features with the inclusion of more features gradually over time in the name of the release of new versions of the same product. This indeed is more revenue for the software vendors but causes regular budgeting for software costs by users. Emanating from the above objectives, the questions are:

1. What are the factors that contributed most to piracy of software in Nigeria.
2. What are the key challenges involved in combating software piracy?
3. What are the financial losses due to software piracy activities?
4. What are the causes of Piracy in Nigeria
5. What are the significant of encryptions on software piracy
6. **Research Hypothesis**

The hypotheses to be tested in this study are;

1. Developer encryption of software has no effect on software piracy
2. Unemployment has no effect on the growth of Piracy in Nigeria
3. Price of software has great effect on the choice of Piracy in Nigeria
4. Government Policies on Piracy has no effect on the growth of Piracy in Nigeria

### Significance of the Study

The proposed study has Educational, Technological and Economic significance. It is of educational significance to students, IT instructors and lecturers in IT field who can use the study as an instructional reference for self-study and tutorials when referring to software, piracy, coding and also to ICT-driven services. It is also of educational significance to researchers in areas of software piracy as a review study. Technologically, it is of significance to software developers who engage in seeking new systems of preventing software piracy through technical means especially means relating to Code encryption technique. The study will also be useful technologically and economically to software vendors who need to identify usage and control installation limits of software as a way of frustrating piracy and preventing financial losses.

**Limitations of the Study**

Little or Non availability of Software Professionals foe the Electronic Questonbaire

Lack of capitals for immediate distributions and return of questionnaire.

### Scope of the Study

This study is focus on preventing software piracy through technology. This technology is the use of a Code Encryption Techniques. The technique has a pre-stored hash value of any software it is integrated to. Every time the software is accessed, it computes a hash value and compares it with the existing pre-stored hash value. If both hash values are the same, the software runs otherwise it terminates. Apart from this feature, it also converts all characters of the serial number to hexadecimal or mashed data so it is very difficult to know the serial number needed to access the software.

## Chapters Organizations

Chapter 1 introduces the research problem and the research approach. Chapter 2 provides an extensive review of previous work and identifies open questions. Chapter 3 provides the Methodology and an in-depth investigation on Combating Software Piracy using Code Encryption Techniques. And how the current state can be improved. Chapter 4 combines the insights of Chapter 2 and Chapter 3 and provides the design and implementation of Combating Software Piracy using code Encryption Techniques. Finally, Chapter 5 concludes the thesis.

### 1.4 Operational Definition of the Terms

**Activation Code:** Some software needs an activation code to unlock it for use. Without activation code, such software cannot be functional (Bahar, 2011).

**Algorithm:** A step-by-step method of solving a problem or achieving a task.

**Amnesty:** a period during which a law is suspended to allow offenders to admit their crime without fear of prosecution.

**Auction Sites:** Online auction sites are like giant flea markets; they provide a forum for buyers and sellers to find or sell just about anything. Auction is also held on the internet. Moores, *et al* (2019)

**BitTorrent networks:** A set of rules that governs how large amount of files can be distributed over a network or over the internet (Kigerl, 2013).

**BSA (Business Software Alliance):** The Business Software Alliance (BSA) is a watchdog group dedicated to fighting software piracy, educating computer users about software copyrights and cyber security, and advocating public policy for electronic commerce, international trade, intellectual property protection, export controls, and emerging technology issues (BSA Global Advocacy and Europe Members, 2013).

**Code Encryption:** This makes the code unreadable by a third party (Bender, *et al* 2011). Passwords are often encrypted and appear as bullets.

**Copyrights:** gives the creator of a work or software exclusive right of distribution of the work or software.

**Copyright Infringement:** The republishing or use of works under copyright without permission from the rightful owner.

**Cyber Crime:** Criminal acts dealing with Computers and Networks, called hacking.

**Encapsulation:** The inclusion of one thing within another thing so that the included thing is not apparent.

**Illegitimate users:** Illegitimate users obtain software from the Software pirates because they want it cheap or even at no pay.

**IPR (Intellectual Property):** Granting a rightful owner of a product certain exclusive right to his product. Third party usage must be under owner’s consent (Apple-Legal-Piracy Policy, October 2013).

**Legitimate users:** Legitimate users purchase software from the Software Vendor, and obtain updates from the same.

**Malicious:** When a software product is pirated, it will contain codes that can cause damage or hurt the software in general.

**Malware:** A software that is hostile and intrusive.

**Module:** A Software program can be broken down into modules (Davidson and Myhrvold, 1996). Each module has its own separate task.

**Obfuscation:** A software programmer can make his codes unreadable by others (Bennett and Layard, 2015).

**ODRL (Open Digital Rights Language):** Is a rights expression language developed to express rights, rules, and conditions - including permissions, prohibitions, obligations, and assertions - for interacting with online content (Iannella, 2001).

**Reseller:** A company or an individual that buys goods in order to sell them to others rather than using them.

**Shareware:** A software vendor can give out his software for free to the public asking them to try it and pay if they like (Merkle, 1993).

**Software Cracking:** Process of illegally making software functional by bypassing its normal process of activation.

**Software developers:** people who create software with all its features.

**Software Installation:** Software cannot function on its own. Computer is required to run software, thus the software is made to run on a computer or machine by installing it there.

**Software Licensing:** when software is being purchased, a license to that software is being issued by the software vendor. Thus procession and use of a software product without its license is an illegal act and considered a crime.

**Software Patents:** a legal monopoly granted for the use, manufacture, and sale of an invention (Nichols, 1999).

**Software Pirates:** The Software Pirates illegally obtain software sold by the Software Vendor, and redistributes (potentially altered) copies of the software causing financial harm to the Software Vendor.

**Software Updates:** Periodically, every software need to be updated with the latest fixes, bugs, patches and security features.

**Software Vendor:** The Software Vendor sells software, keeps a list of registered users, and ensures that the legitimate users’ software are updated when due.

**Watermarking:** one can embed watermarks on an image or software in order to be able to trace it (Myles and Collberg, 2003).

**XML:** Extensible Mark-up Language. A computer language used to encode a document to be understood by both humans and machine or compute

## REFERENCES

Howard, M., & LeBlanc, D. (2003). *Writing secure code*. Pearson Education.

Steinmetz, K. F., & Tunnell, K. D. (2013). Under the pixelated jolly roger: A study of on-line pirates. *Deviant Behavior*, *34*(1), 53-67.

Moores, T. T., & Esichaikul, V. (2011). Socialization and software piracy: A study. *Journal of Computer Information Systems*, *51*(3), 1-9.

Warner, R., & Sloan, R. H. (2012). Vulnerable Software: Product-Risk Norms and the Problem of Unauthorized Access. *U. Ill. JL Tech. & Pol'y*, 45.

Bishop, J. (2005). Building international empires of sound: concentrations of power and property in the “global” music market. *Popular Music and Society*, *28*(4), 443-471.

Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., ... & Savage, S. (2013). Measuring the cost of cybercrime. In *The economics of information security and privacy* (pp. 265-300). Springer, Berlin, Heidelberg.

Bahar, R. (2011). *U.S. Patent Application No. 13/196,158*.

Moores, T. T., Nill, A., & Rothenberger, M. A. (2009). Knowledge of software piracy as an antecedent to reducing pirating behavior. *Journal of Computer Information Systems*, *50*(1), 82-89.

Kigerl, A. C. (2013). Infringing nations: Predicting software piracy rates, bittorrent tracker hosting, and p2p file sharing client downloads between countries. *International Journal of Cyber Criminology*, *7*(1), 62.

Bender, B. H., Cronin, G. J., & Lloyd, J. A. (2011). *U.S. Patent No. 7,930,252*. Washington, DC: U.S. Patent and Trademark Office.

Davidson, R. I., & Myhrvold, N. (1996). *U.S. Patent No. 5,559,884*. Washington, DC: U.S. Patent and Trademark Office.

Bennett, L., & Layard, A. (2015). Legal geography: Becoming spatial detectives. *Geography Compass*, *9*(7), 406-422.

Iannella, R. (2001). ODRL: The open digital rights language initiative.

Merkle, R. C. (2001). *U.S. Patent No. 6,330,549*. Washington, DC: U.S. Patent and Trademark Office.

Nichols, K. (1999). The age of software patents. *Computer*, *32*(4), 25-31.

Myles, G., & Collberg, C. (2003, November). Software watermarking through register allocation: Implementation, analysis, and attacks. In *International Conference on Information Security and Cryptology* (pp. 274-293). Springer, Berlin, Heidelberg.

### CHAPTER TWO

### LITERATURE REVIEW

### 2.1 Introduction

In the course of this research, related works on various previous techniques applied in the prevention of software piracy are studied and reviewed in an attempt to justify its relevance and contribution within the frontiers of knowledge. Software piracy is a behavioral act, meaning that it is a deed that most people who engage in tend to be conscious of. These pirates have their different reasons for engaging in software piracy, seems not ready to quit the act as long as new software are continuously released. This is the main point of view that this research is established on for reviewing these literatures. Most of the techniques that have been employed in the past to combat software piracy only succeeded in making piracy a bit more difficult for pirates but they eventually break through at last. A+- perfect technique for piracy control could eventually lead to causing legitimate users so much inconvenience which is not the goal of software vendors. Piracy controls usually take account of the software manufacturer’s desire to cause legitimate users as little inconvenience as possible (Cronin, 2002).

Software vendors and manufacturers has deployed several piracy control measures in which most of these measures are still in place and are often practiced in a way of combining them in order to achieve a more effective piracy control. Some of these control measures are forceful measures while some others are geared towards shaping the mind sets of various software users to voluntarily abstain from the act of piracy. Software pirates often neglect these forceful measures which come in several forms of copyright infringement laws with their associated copyright infringement penalties. Also talking down most of these pirates using moral-based control measures has also helped. Technical measures were later adopted by software manufacturers but with several limitations usually pointing towards acts of professional programmers who also engage in software piracy.

According to (Korhonen,2015). Piracy controls may be ethical (moral-based), legal or technical. Until today, there are many researches still carried out on piracy control/prevention, but more often on the technical controls, since it seems to be the only measure that at least exempts non-programmers from the direct act of piracy. Some software vendors have also adopted a piracy control measure of seeking to hear from the pirates of their software in a way to find out reasons they download their software products without paying for them.

*“There was a point to all this, and it was partly to sell more (I have bills to pay!) as well as hopefully get more people to legitimately play my games. I'd be very happy if some reduction of overall piracy happened too, as I love PC gaming and the current situation is only helping to kill it off. I've thought hard about everything people have said and I have decided to change a few things about my games.”* (Cliff, *et al* 2011).

Cliff also continued in his conclusion, *“…but I think now I know more about why pirates do what they do, I'll be in a better position to keep doing what I wanted, which is making games for the PC.”* Each of these piracy control measures has no doubt contributed its quota to the control of software piracy. The BSA (2013) has listed resources available for learning more about preventing software piracy. These resources are through a number of industry organizations dedicated to this cause, as well as government sites that describes intellectual property rights, protections, and penalties for abuse. These industry organizations are:

**BSA:** The Business Software Alliance is a global organization formed to advance free and open world transfer for legitimate software by advocating strong intellectual property protection. The BSA has Global membership levels, Global Advocacy membership level, Regional membership level, Regional Anti- piracy Membership level, Regional Advocacy membership level and the Country membership level. Members of the Business Software Alliance (BSA Global, Global Advocacy and Europe Members, 2013) are the following:

* ACCA Software
* Adobe
* Altium Limited
* ANSYS, Inc.
* Apple
* Autodata Limited
* Autodesk
* AVG
* Bentley Systems
* CA Technologies
* CG Tech Ltd
* CNC Software – Mastercam
* Corel
* Dell
* IBM
* Intel
* Intuit
* McAfee
* Microsoft
* Minitab
* NetCadUlusal CAD
* Oracle
* PTC
* RockwellAutomation
* Rosetta Stone
* Siemens PLM software, Inc.
* Symantec
* Tekla
* The MathWorks

**CAAST:** The Canadian Alliance against Software Theft. This is an industry alliance of software manufacturers that share the common goal of reducing software piracy. CAAST provides educational information to corporations, consumers, academic institutions, and resellers about software theft and its implications. CAAST works with BSA.

**SIIA:** The Software and Information Industry Association. This is a trade organization for the software and digital content industry. SIIA offers global services in business development, government relations, corporate education, and intellectual property protection to companies. It also advocates a legal and regulatory environment that benefits the entire industry. The SIIA has the following principal mission:

* + - **Promote the Industry:** SIIA promotes the common interests of the software and digital content industry as a whole, as well as its component parts.
    - **Protect the Industry:** SIIA protects the intellectual property of member companies, and advocates a legal and regulatory environment that benefits the entire industry.
    - **Inform the Industry:** SIIA informs the industry and the broader public by serving as a resource on trends, technologies, policies and related issues that affect member firms and demonstrate the contribution of the industry to the broader economy.
    - **FAST:** The Federation against Software Theft was set up in 1984 by the British Computer Society's Copyright Committee, and was the first software copyright organization in the U.K. Its first action was to raise the awareness of software piracy and to lobby the U.K. Parliament for changes in the Copyright Act of 1956 to reflect the needs of software authors and publishers. This campaign was successful, and FAST has since been able to influence other legislation that impacts the proper safeguarding of software. The work of FAST in this area has directly influenced the way software copyright law and investigations are carried out in many other countries. review on the study of piracy control measures are conducted in this course of research.
  1. **Methods of Protecting Software**

There are three major types of software threats: Reverse Engineering, Tampering, and Piracy. Code encryption, Obfuscation, Tamperproofing and Watermarking are the respective techniques used to curb the existing threats (Collberg and Thomborson, 2002). Software reverse engineering is a technique to inspect the inner workings of software applications, tampering covers ways to modify a software while piracy concerns unauthorized use of software.

* + - The organization of this review shall be sequenced in the order: ethical measures, legal measures and technical measures. But this review shall have a broader concentration on the technical control measures than the others.
  1. **Piracy Control Using Ethical Measures**

Cronin, (2002). suggests that the “right to fair compensation” is a basic principle of “universal” ethics. The essence of employing ethical measures in controlling software piracy is to make software piracy morally unappealing. So many arguments have followed that piracy is morally wrong and should be avoided. Cronin, (2001). has listed the ethical measures of controlling software piracy: Shareware, Amnestyand Educational/Appeal.

## 2.3.1 Shareware: This is also referred to as “trialware” or “demoware”. It is proprietary software that is available free of charge on a trial basis, usually with the condition that users pay a fee for continued use and support. Around 1982, Bob Wallace PC-Write, a word processor and called it shareware. Later, it was established as a software marketing method. According to Wikipedia, shareware was the only economical way for independent software authors to get their products onto users’ desktops. This measure was also later put in place to make up for the excuse of people who download software products without paying for them claiming they only wanted to test the software. It gives them the opportunity to use the product for a given period and then pay for it if they like it. Shareware is usually offered either with certain features only available after the license is purchased, or as a full version but for a limited trial period of time. Shareware has been an option put in place for combating software piracy, though it does not directly stop software piracy, it is just dependent on the honesty of the customer.

## Amnesty – as a Piracy Control Measure

The amnesty measure of software piracy control encourages people possess illegitimate or pirated software to surrender them with no risk of prosecution Mykhalchenko and Wiegratz, (2019). posted on the internet that a 30-days Software piracy amnesty campaign was launched in Kenya in 2007 by the Kenya Copyright Board – KeCoBo, with the support of the Microsoft East and South Africa encouraging end-users running counterfeit software to “come clean” by discontinuing illegal use of software and acquiring genuine versions without penalty. With the new amnesty campaign, the focus was primarily on creating awareness about the prevalence of software piracy in Kenya. A “clean dealer” program was launched by Microsoft which seeks to provide all users with a list of authorised dealers and resellers of Microsoft software in Kenya. The “clean dealers” were facilitating the exchange of counterfeit software for genuine Microsoft software and educate users on the overwhelming number of “high quality” counterfeit software in the market.

According to the Business Software Alliance (BSA) 2010 – 2011 Report, the level of piracy in Kenya dropped since 2007 – 2011 from 83% to 79%. A report from Mykhalchenko and Wiegratz, (2019). Said that AutoDesk offered users of their pirated software products to approach local Autodesk channel partners to purchase legal versions of AutoCAD and related software products without fear of retroactive prosecution. This amnesty program was supported by local Kenyan authorities and stakeholders like the Kenya Copyright Board. It was aimed at not only to drastically reduce software piracy in the region to protect the local economy, but also to educate their customers on the protection from the effects of malicious coding that is often contained in pirated versions.

## Education /Appeal – As a Piracy Control Measure

Software users are often educated to understand that purchasing software is the only fair compensation for the authors and distributors of that software. ‘APPEAL’ is an ethical piracy control measure in which a public campaign is carried out with an attempt to persuade software pirates, software distributors or software end-users to say NO to piracy or use of pirated software. The BSA fund frequent advertising campaigns in an attempt to steer public thinking towards the view that software piracy is an illegal and economically damaging activity. These campaigns are in the form of letters to legislators and prominent newspapers, and paid print and media advertising (Gates and Kelly, 2002).

This ethical measure is all about changing peoples’ mindset about software piracy. An appeal to users to avoid software piracy and copyright infringements goes along with a strategy of sensitisation to end users or educating them on how best to completely avoid the act. Collard *et al*, (2015). created seven steps on how to avoid copy infringements. These includes: understanding the scope of copyright law, avoid taking things from the internet because they are almost always copyrighted by default, don’t confuse copyright and other forms of intellectual property, be creative, learn about public domain laws for your jurisdiction, don’t rely on “fair dealings” in many jurisdictions, and finally be wary about writing fiction based on other works. These educational measures help to grant the necessary sensitization on end users on believing they can willingly surrender to abstaining from software piracy. Moores, *et al* (2009). has shown the efforts of the BSA to educate the people on software piracy and its effects on society.

## Piracy Control Using Legal Measures

It is very important to note that pirated software is illegal and a crime in most countries. Many agencies and organizations have been created for the sole purpose of preventing the activities of software pirates. These agencies and organizations create laws which are used to enforce software piracy prevention and these laws prevent piracy by creating a fear of consequences that being caught pirating brings. A new police unit aimed at targeting intellectual property crime has was announced by the intellectual Property Minister, Lord younger and City of London Police Commissioner, AdrainLeppard on 28th June, 2013 Jason-Lloyd, (2013). They aimed at creating an independent police unit that will coordinate the national and international response from law enforcement and public and private sector partners so as to safeguard jobs and protecting peoples’ personal and computer safety by ensuring that they are not exposed to counterfeit goods and unauthorized copyright content.

In 2005, end user licensing agreements were created. Before you register your copy of any software, you are required to indicate that you have read and understand the EULA. Since there are no guidelines, restrictions, or standards for the content or language of EULAs, companies can include a statement to the effect that the end-user is solely responsible for any illegal activities and the consumption or exchange of copyrighted materials. Remember that computers and the internet are tools that can be used for any reason good, bad, or deceptive. Whether you download a copy, or take a copy from the retail store, failure to pay for any item or comply with end user licensing agreements is theft (Orme, 2014).

Today, when the software is being purchased, a license to that software is being issued by the software vendor. Thus procession and use of a software product without its license is an illegal act and considered a crime. Copying, distributing and exchanging a software product with other people or on the internet is a violation of its license and is a violation of copyright law. Even when amnesty was granted by the Microsoft to Kenya end users that users of pirated software should surrender them without any fear of prosecution, the amnesty campaign only lasted for 30 days after which local authorities were used to track down organisations and individuals who were still in possession of pirated software for prosecution (Mach and Nash, 2019).

In September 2013, the Spanish government has also approved new laws, meaning that owners of websites that link to pirated content will be jailed for up to six years and this measure will not take effect until January, 2014 (Law,2013). Spain last created an anti-piracy law in 2011, which helped it for a while — until the situation turned for the worse again. RojaDirecta.com, for instance, linked to unauthorized sports streams, was shut down by the U.S., but relaunched as RojaDirecta.me, based in Montenegro Clever. Previously, Spanish authorities could only punish the actual users the ones who copied and distributed copyrighted material. It was unable to pursue the actual Pirate Bay-like sites. Now, under the new regulation, the actual users themselves will not face any punishment. Oddly, peer-to-peer file-sharing platforms and search engines are exempt from the rules and won’t face legal action, interesting.

According to Professor of Law (McMullen, 2014). The intended purpose of the Digital Millennium Copyright Act (DMCA), an act signed into law in 1998 as the first major legislative attempt at thwarting piracy, was to protect copyright holders without hindering the users of copyrighted works. The first major issued related to the DMCA came about when a person on the Internet who called himself AiboPet hacked the Sony Corporation- produced Aibo and developed enhancements for the robotic toy, which he then provided to other Aibo users via the Internet. Sony claimed that the enhancements were illegal according to the DMCA and wanted AiboPet to pull all the enhancement programs off the Internet. Fortunately for AiboPet, the many people who enjoyed the enhancements were able to convince Sony to back off. Still, the DMCA had almost been the cause of a developer-consumer conflict over programs that should have been protected by fair use rights, especially considering the programs did not harm Sony’s sales or profit AiboPet in any way.

Unfortunately, the DMCA has since been used many times to hinder fair use because the act considers illegal any circumvention of technological protection measures (TPMs), even if such circumvention is not committed with the intent to harm the developer of the copyrighted product. Legal measures prevent piracy by creating a fear of the consequences that being caught pirating will bring. Legal systems in the United Kingdom, the United States of America, and countries that have inherited much of their legal systems from these two countries and their earlier origins provide legislation to allow prosecution of pirates through copyright, software patents (Nichols, 1999). and software licensing. To impose a fear of consequences, there must be some form of liability for the act of software piracy and this liability must be able to be proven in a court of law. The liability may rest with the pirate, the provider of the distribution channel, the end- user, or a combination of these Litman, (1996).

Historically, copyright evolved out of the printing revolution that followed Gutenberg’s invention of the printing press. Copyright law is based almost entirely around this “print paradigm” and its associated concepts of the permanence of a publication and its repeated availability (e.g. a book once purchased may be read many times) Harvey, (2002). The distinction between copyright and patents is an important one. In U.S. law a patent is a “legal monopoly granted for the manufacture, and sale of an invention”. On the other hand, copyright “applies to a particular, tangible piece of work” (Nichols, 1999). Copyright protection is automatic in the U.S. and in New Zealand. It affords the author of a work control over that work and gives them the right to transfer that ownership to another party.

A breach of copyright is addressed in the courts, where the plaintiff must prove their copyright over the work in question. A patent holder’s monopoly means they can prevent others from using their patented device, even if others invent it independently (Nichols, 1999). Copyright and software patents can be considered an effective legal protection where the offender and manufacturer are located in the same country and are therefore subject to the same intellectual property laws. Difficulties arise however where the parties are subject to different legal jurisdictions. The Business Software Alliance (BSA), an affiliation of leading (mostly American) software companies fund surveys of international piracy and acts as a lobby group in 5 bringing political pressure against countries who do not cooperate with international copyright agreements. The piracy industry in China was virtually closed down due to threats of trade sanctions (Dakin, 1997).

The BSA’s most recent global piracy survey found that the highest rates of piracy occur in Asian and Middle Eastern countries, so the effectiveness of legal piracy prevention is dependent entirely on these countries cooperating with international copyright agreements Bagchi,*et al* (2006)Software licences are essentially a contract between the user(s) of a given item of software and the manufacturer or distributor. Most commercial software requires some form of acknowledgement from the user that they have read and understood the terms of the accompanying licence and agree to abide by these terms when they use the software. Many types of licensing are available, ranging from “site licences” that allow all users in a given geographical location or set of network addresses to use copies of software, to simple single-user agreements that prevent any duplication at all Allaire, (2002). Some software licenses include clauses that require the owner of the software licence to submit to regular audits of their premises to determine their compliance with the licence conditions. While such clauses are not usually found in single end-user licences, an attempt to do so was recently made. Inprise – the manufacturers of the Borland brand of development products Cronin, (2002). Inserted a compulsory audit clause in the single-user licence for JBuilder 5 and Kylix 2. After much protest from the user communities of these products the clause was withdrawn Cronin, (2002).

* 1. **Piracy Control Using Technical Measures**

There have been so many technical measures put in place to combat software in the past. Software vendors and manufacturers are continuously investing a great deal of resources which includes sufficient time on technical measures for the total prevention of software piracy.

Stalberg and Cronin, (2011). sub-categorised technical control measures of preventing software piracy into the following: First are those controls that act to prevent the duplication of software, examples are code obfuscation, encryption, simple checks, et cetera. Secondly, those controls that increases the likelihood of offenders being cut and prosecuted like observation and watermarking. Some of these technical measures for preventing software piracy are the following:

## 2.5.1 Method of Preventing Software Piracy during Installation from a Read-Only Storage Medium

Larsson *et al,* (2001). researched on an invention of a method for preventing software piracy during installation from a Read-Only Storage Medium. The invention was directed to a method and system for limiting the number of installations of a computer software program located on a “read only” disk from the read only disk to a computer. In the preferred embodiment, the invention may be incorporated into an anti-piracy system such as Acme 3.0, which is marketed by Microsoft Corporation of Redmond, Wash. Furthermore, the invention may also be referred to as “Hardware Detection during Install” or “HDDI”. Briefly described, the invention limits the number of installations of a computer software program stored on a read only device such as a compact disk (CD), from the CD to one or more computers. In an exemplary embodiment, computers have a permanent read/write storage device (such as a hard drive), an optical disk drive for the CD, and a removable read/write storage memory (such as a floppy disk).

The CD is linked to the floppy disk by information stored on the floppy disk corresponding to information on the CD. After the link between the CD and floppy disk is verified, installation of the software from the CD to the hard drive of the computer may be accomplished. More specifically, in response to the initiation of software installation from the CD to the computer, the user is requested to insert the floppy disk into the computer. The method of the invention first determines if the inserted floppy disk is properly linked to the CD.

Once the proper link has been verified, the method for the invention determines whether the installation of the computer software from the CD to the computer is authorized by comparing identification information of the computer to identification information of other computers on which the software has been previously installed. If the installation is authorized, installation of the software to the computer is allowed. If the installation is not authorized, the installation is disallowed. Asongu, *et al* (2018). argued that despite all the efforts intended to prevent software piracy by the application of this method, major flaws are still noted. In today’s technological advancement, present computer systems has no floppy disk drives created with them, rather the CD drives are used and is viewed to be more acceptable by all users of the computers due to the fact that running software programs on floppy disk is slow.

Also there is a creation of the term “dependency” between the two storage media, in the sense that without one medium the installation of the software to the computer system is not accomplished. A floppy disk referred to as license floppy will be required if the user initialized the installation of the software, and if such licensed floppy is not inserted the installation is disallowed.

## The Online Auction Tracking System (Oats)

Another remarkable technical means of preventing software piracy employed so far is the Online Auction Tracking System by the BSA Yang, *et al* (2009). The Online Auction Tracking System (OATS) is a proprietary tool that monitors auction sites and BitTorrent networks on a continuous basis, while another tool monitors other P2P activity. This system helps the BSA to identify offerings of illegal software via various websites and P2P networks. This will lead to the BSA giving a “Take down” notice on the ISPs of the websites in a request for them to remove pirated software. In the first half of 2009, BSA stepped up its efforts in this area and issued almost 2.4 million takedown notices related to P2P and BitTorrent file sharing, an increase of more than 200 percent over the same period in 2008.

In 2007, BSA launched an in-house Internet “crawler” to strike further up the BitTorrent supply chain, in addition to the notices sent at the “demand” level where permitted by law. In the first half of 2009, BSA more than doubled its impact with this tool compared to that of 2008, requesting the removal of almost 103,000 torrent files from nine of the largest BitTorrent index sites worldwide. These torrent files were being used by nearly 2.9 million individuals to download software with a retail value of more than $974 million. When BSA finds suspicious software being offered on auction sites, it issues takedown requests to the auction site providers to remove those listings. During the first half of 2009, BSA has expanded its efforts in this area as well, requesting auction-site providers to shut down more than 19,000 auctions offering about 128,000 products worth a combined $55 million.

## Activation Code System of Preventing Software Piracy

According to a publication by Bahar, (2011). the activation code system and method of piracy prevention is a system where a given software application limits the number of times that such software application is activated. The user is expected to provide a unique software identification code, relating to the software which the user is trying to install. This is directed to a remote server which determines the number of times that specific software has already been activated. It then provides an activation code to the user if the predefined threshold has not been exceeded which will activate the installed software and makes it usable. But if the predefined threshold has been exceeded, the system will suspect that the user is a pirate and will not send an activation code. In the detailed description of the invention, Bahar described that the user will first install the software which will expectedly install successfully.

When the user wants to access or use the software for the first time, the software will display a message telling the user that activation for the software is needed. Of course the user would want the installed software activated, hence follows the activation wizard. But at this point, it is only the non-pirate user that can continue from here since the activation panel will request for the identification code of the application software. This identification code would consist of an elongated alphanumeric code sequence, such as “program file(s)”, it is preferred that it consists of a short code sequence of alphanumeric character, e.g. XGY-U75K-RD3P1. When the user keys in the identification code, the software connects to a remote server in order to verify the identification code. After this verification, the remote server is expected to send an activation code to the user, but his will only be if that specific application software has not exceeded a predefined threshold limit by the application software vendor.

This activation process is applicable over a communication network like the internet, hardwire telephony, wireless networks including cellular and PCS systems, satellite networks, localized and regional networks such as intranets and local area networks (LANs) which interconnect a relatively few number of systems or terminals, typically by a means of a centralised server. Bahar (2011) also described in this invention that the remote server will collect users’ information during the activation process. This could be manual information provided by the user which includes and not limited to the user’s personal identification information such as username, address, location, phone number, et cetera. This could also be information relating to software collected automatically by the remote server and stored in its storage element. This information includes and not limited to identification code, a product serial number, model number, and name and/or version number. The currently transmitted data is compared with the previously archived data in the storage element of the remote service system. This will allow the remote service system know if the user is trying to install a pirated version of the specific software. If the archived data establishes that the software is legally registered to a completely distinct user, such may indicate that the user currently online is trying to activate a pirated version of that specific application software.

This result will occur if the archived data referencing the software does not match the user data currently being transmitted by the user system, and/or if the archived data indicates that there have been repeated and numerous attempts to activate the same software. Bahar also stated that with a multiple activation attempts on one specific software product registered in the remote service system, this may indicate a potential piracy where the specific software product is duplicated and shared amongst multitude of users. The software vendor will often contact the original registered user in order to investigate potential piracy. This user(s) may also be blacklisted, with reference to the software identification code, in order to prohibit the specific application software from receiving future activation codes from the remote service system which will render the identified software void and permanently dysfunctional.

One of the main advantages of using a unique activation code is the drastic curtailment of software piracy. Each authorized copy of software application is designed to be responsive to a distinct activation code. As such, an attempt to pirate distinct software applications would entail a tedious and time consuming task requiring the hacker to uncover the activation code of each individual, authorized software product. Furthermore, a unique activation code will not allow for the activation of any “general” copy of the software which would otherwise be responsive to a common activation code. As an alternative to a unique activation code, a common activation code would activate all “same type” software applications. Developing “same type” software to be responsive to a common activation code may be advantageous given the potential for reducing confusion and troubleshooting errors which could arise during the software manufacturing and online activation stages.

Activation code preferably remains undisclosed to user. The need for the activation code will compel user to register the software online with the remote service system. Furthermore, and more importantly, having the activation code only known to the remote service provider and its business affiliates (such as the software manufacturer) will prevent piracy of the software. This is because users who wish to pirate the software will not be able to replicate the activation code and distribute it along with a medium (e.g. CD ROM) containing a copy of the software. Activation code is preferably designed to be immune from discovery by computer hackers and sophisticated programmers.

The objective is to prevent these individuals from “breaking in” to the software and either re- writing or discovering the undisclosed activation code. This may require constructing the activation code as a long code sequence which results in a program file(s) because using a long code sequence to generate activation codes makes the activation codes more difficult to replicate than using a short code sequence. Additionally, other measures may include code encryption as well as any other programming methods known to those skilled in the relevant technical art.

An example of an activation code system is the ALK technologies Software Licensing. Andrew, (2013). ALK Software (the "**Software**") includes an activation based software licensing scheme both for base License products and upgrade features. The Software is delivered with a Product Key Code of 25 digits ("**PKC**")\*, which corresponds to the purchased features. This PKC coupled with a license number (the "**License Number**"), created by the Software when it first runs on your Device (the "**Device**") or platform (the "**Platform**") generates an activation code which will unlock your Software for normal use. It allows either automatic or manual ways of software activation. It also allows transferring or deactivating your ALK Software License but can only be possible between devices with the same operating system, for example, between Android to Android or iPhone to iPhone.

The Software may include product activation and other technology to prevent unauthorized use and copying. When provided with a Product Key Code, Google order number or any similar mechanism, you will need to activate your Software with the associated method in order to use it. If you try to activate an excessive number of times, the anti Piracy Protection may cause your Software to lock and prevent you from further activating your Software Andrew, (2013).

## Generating and Auditing a Signature for Computer Program

According to an invention published by Davidson and Myhrvold (1996), which involves using generated signatures for preventing software piracy. The signatures are similar to serial numbers in that they uniquely identify each copy of the executable program, but different in that a signature is an intrinsic part of the executable module program, while the serial number is an extrinsic data item. Computer programs are typically sold in the form of executable modules. An executable module contains instructions that have been compiled and translated into machine code so that they can be loaded into memory and executed.

After software manufacturers create an executable module, they make multiple copies of the executable module and then sell the copies to purchasers. While making the copies, software manufacturers typically insert a unique serial number at a predetermined location within each copy of the executable module. The serial number is a unit of data that distinguishes each copy of the executable module. If a subsequent copy is made of the purchased copy, the subsequent copy should bear the same serial number as the purchased copy. When a purchaser purchases a copy of the executable module, the purchaser's name and the serial number of the purchased copy are typically registered with the software manufacturer.

This registration may be accomplished by the purchaser submitting a registration form to the software manufacturer or, in some instances, the software manufacturer or a retail vendor recording the necessary information at the time of purchase. By associating a serial number with each purchaser, software manufacturers hope to deter unauthorized copying. By comparing the serial number of a copy in question with the software manufacturer's list of recorded serial numbers, the software manufacturer may determine whether the copy in question is an authorized copy. If the software manufacturer determines that the copy in question is an unauthorized copy, the serial number within the unauthorized copy may indicate from which authorized copy the unauthorized copy was derived. Unfortunately, the purchaser (or someone who has access to the authorized copy) may easily remove or alter the serial number, make multiple unauthorized copies, and then distribute or even sell the unauthorized copies to third parties.

The software manufacturer is able to identify such copies as unauthorized, but, without a valid serial number to rely on, the software manufacturer is unable to determine the origin of the unauthorized copies. Similarly, when a purchaser fails to register the serial number of a purchased copy with the software manufacturer, the software manufacturer may later be unable to determine whether the serial number of the purchased copy has been altered. Software manufacturers require a method of including a serial number or other identifying mark within a copy of an executable module in such a way that the serial number or identifying mark cannot be easily removed or altered. Additionally, software manufacturers require a method of analyzing a copy of an executable module to determine its serial number or identifying mark.

## Code Obfuscation Techniques of Preventing Software Piracy

Cappaert, (2012). described *Code obfuscation* as a set of program transformations that make program code and/or program execution difficult to analyze. He continued that at first, obfuscation hinders manual inspection of program internals. By renaming variables and functions, and breaking down structures, it protects against reverse- engineering. It protects both storage and usage of keys, and it can hide certain properties such as a software fingerprint or a watermark, or even the location of a flaw in the case of an obfuscated patch.

However, code obfuscation itself does not protect from code lifting or software piracy. It merely strengthens built-in protection mechanisms, e.g. against tampering or piracy. When a programmer spends hours writing code, its only logical that such code be patented. But copyright laws and patent laws (uspto.gov) do not serve all the needs of a programmer. One way of preventing plagiarism and hacking is to employ code obfuscation Crusoe, (2019). Crusoe also stated that some languages are more prone to obfuscation than others and are most often cited as easily "obfuscatable" languages. Macro pre- processors are often used to create hard to read code by masking the standard language syntax and grammar from the main body of code. C, C++ and Pearl Programming languages are some examples of languages easy to obfuscate.

Types of Obfuscations include simple keyword substitution, use or non-use of whitespace to create artistic effects, and self-generating or heavily compressed programs. A variety of tools exists to perform or assist with code obfuscation. These include experimental research tools created by academics, hobbyist tools, commercial products written by professionals, and open-source software. ABCME is an obfuscation tool that assimilates assembly code to software codes. There also exist de-obfuscation tools that attempt to perform the reverse transformation. .

According to Mateas and Montfort, (2003). in 1984 Landon Curt Noll and Larry Bassel held the first International Obfuscated C Code Contest. The contest was a success that has been repeated many times; judging of the 18th IOCCC was underway when this article was written. Only small, complete C programs can be entered in the contest, which rewards originality and the aesthetic abuse of the C language. The contest's stated goals include demonstrating the importance of programming style (“in an ironic way”) and illustrating “some of the subtleties of the C language.” An anonymous entry in the first IOCCC accomplishes these goals in only two lines, and also plays on the conventional “hello,

world!” program, a program which is typically used as a simple first example when learning a programming language.

At best, obfuscation merely makes it time-consuming, but not impossible, to reverse engineer a program. However, some developers may employ code obfuscation for the purpose of reducing file size or increasing security.

## Watermarking and Fingerprinting Algorithms of Preventing Software Piracy

Myles and Collberg, (2003). have described watermarking as a technique used to aid in the prevention of software piracy by embedding a copyright notice on the software that can be used to provide proof of ownership. Software watermarking algorithms can be used to embed a customer identification number (a fingerprint) into a Java program in order to trace software pirates. This has helped to discourage illegal copying and redistribution of software and also acts as a fingerprint that can be used to trace the source of illegal redistribution. Myles and Collberg has stressed that watermarking does not prevent (but rather discourages) illegal software copying and redistribution. They also gave instructions on how to watermark software: insert new (non-functional and non-executed) code, re-order code where it does not change the functionality and manipulate instruction frequencies. Fingerprinting is similar to watermarking, except a different secret message is embedded in every distributed cover message. This may allow us not only to detect when theft has occurred, but also to trace the software copyright violator. A watermark can prove software ownership but it cannot point to the actual culprit of the illicit action. To add such characteristics to the watermark, it becomes a fingerprint: in fact, the fingerprint data is a watermark containing data from the individual customer.

A typical fingerprint includes a vendor, product, and customer identification numbers. Small, (2010). published that Developers watermark software by implanting unique identifiers-**called the watermark**-into software in order to assert ownership to the rightful owner but also to track down the pirate after the illicit act. So, software watermarking itself does also prevent piracy by scaring the user from copyright infringement by increasing the possibility of getting caught. It must be mentioned that watermarking is not a simple technique. Overall, watermarking and fingerprinting are very effective in fighting back software piracy. But the result is not perfect which makes additional protection techniques most advisable. Combined with other known techniques, cracking such schemes becomes much more time consuming that even the experienced cracker may choose to go for easier software to crack Small, (2010).

## Software Piracy Prevention through Software Aging

This technique was designed to rely on periodic updates of software to discourage software piracy. Cronin, (2002). Introduced this novel method of preventing software piracy that intends to benefit legitimate users while inflicting damage on illegitimate users. The updates are to occur, or else the software becomes increasingly useless over time. Through these frequent updates, the pirates who illegally sell these software products will also be pushed to frequently provide the required exact updates as enjoyed by the legitimate users, or else, the software will become incompatible to that of the legitimate users. This frequent reliance of illegitimate users on pirates often leads to discouraging the pirates.

This also increases the operating costs for the pirate, and forces the pirate to keep in touch with its customers (which in turn increases the risk of discovery by authorities). Altogether, Cronin believe that this technique will infringe upon the economical viability of piracy by raising the operating costs for the pirate, and lowering the resale value of his merchandise. At the same time, the costs of maintaining the updates are kept low for the distributor, assuming that the updates are performed using an on-line protocol. This is a reasonable approach given that most computer users also are modem owners. They noted that their methods work even under the pessimistic assumption that the pirate is able to remove or alter any pieces of code and data that are used to detect or prevent piracy, such as code-embedded watermarks, verification of CPU identities, and similar.

Also, in addition to defending against piracy, this suggested model makes software rentals easier to administer (simply by charging for the updates as opposed to providing these for free). It also incorporates a method of forcing interaction between the software distributor and the legitimate users in which end-users are expected to provide their software registration number. If an illegitimate user were to contact the distributor for an update, he would have to give a registration number. In case it is invalid, the distributor may supply a “random update”, thereby efficiently corrupting the operation of the pirated software. In case it is valid, it allows the distributor to partially trace the criminal (from software distribution lists). He may also supply a random update in case he has already updated the software for the given user in this time period (i.e., another clone of it).

Therefore, a rational pirate would have to alter the portion of the software that requests updates in a way that it either contacts the pirate (which would hardwire a contract address for the pirate in every piece of pirated software sold) or that awaits an update from the pirate. If this is not done, then illegitimate users will be refused updates, which will lower the value of the pirated software to them, and therefore also the possible profit to the pirate.Software aging however has its issues regarding making the updates requisition occur more frequently in such a manner that will frustrate the software pirates. Amongst others, the problem of connection interruption between a user and the distributor contributes to the issues faced by this method

## Preventing Software Piracy through Software Diversity

In this technique, each installed copy of a program is unique. According to Djekic and Loebbecke (2007). 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. The scheme also includes software updates to migrate from a static nature of defense to a more dynamic one. In particular, software updates in the scheme was crafted to ensure that they work for one, and only one, installed copy. When updates are no longer provided for installed copies that are known to be illegitimate, a pirate will need to break through a new line of defense with every critical update.

Djekic and Loebbecke, on their piracy prevention scheme, continued that the core of their protection scheme consists of two levels of diversification. At the first level each distributed copy is different. At the second level every installation of a specific copy is different. Each specific copy installed on a specific machine was referred to as an instance, and an instance specific update as a tailored update. An instance must be activated through interaction with the software provider and contains links to the hardware to ensure that an instance cannot simply be copied to another machine. The software provider maintains a database that keeps track of the legitimate instances and their characteristics.

The instances are crafted in such a way that they differ significantly; allowing the creation of updates fit for one instance and one instance only in such a way that a tailored update cannot easily be generalized for other instances. When a user requests an update, he needs to identify the instance he wants to obtain the update for. The software provider then checks if the request is legitimate, looks up the characteristics of the instance and generates a tailored update.

However, this scheme has gained some advantages over other technical means of preventing software piracy since most of these other technical measures have static nature of defense in which a protection mechanism is built into the software and once this protection is broken, no further steps can be taken to protect the intellectual property.

## Summary of Literature Review

Through these various reviews, it is so obvious that the government of various countries, the BSA, legal authorities, programmers, legitimate users of software and software vendors have altogether been working hard to achieve the prevention of software piracy which has been proven to have an effect on each of the listed. Government bodies have been seen to partner with the BSA to fight software piracy since it has affected the expected revenue which should come to them through the payment of tax from the software vendors. Thus, they have sponsored anti-piracy campaigns, worked with the police Olsson and Luchjenbroers (2013). implored legal means of prosecution of software pirates. They have also made use of education as an ethical measure to make software piracy unappealing.

Amnesty has also been a measure that has worked to a great extent by encouraging piracy defaulters and end- users of pirated software products to openly surrender the act without fear of prosecution, once within the given period of amnesty. The software vendors, on the other hand, have not abandoned the fight for software piracy only to the BSA and the government; they have also individually implored so many numerous technical measures as well as other non-technical measures to make software piracy non-profitable to the pirates by cost reduction of software products and the use of shareware.

Jakobbson and Reiter, (2001). has also used software aging to further frustrate the activities of software pirates by a way of enabling frequent automatic updates and affecting a communication between the software vendors and legitimate users thereby forcing a responsibility on the software pirates to periodically update their clients which are illegitimate users as well. Generally, using technology to fight software piracy has gone as far as making software piracy less profitable, making software piracy frustrating to the pirates, making it possible to trace the pirates, disrupt the normal functioning of the pirated software and making them useless over an expected required period of obtaining updates, making it possible for software vendors to monitor how end-users use the products sold to them through DRM, et cetera. Programmers have even implored educational means as well by creating websites where they ask and reply to questions regarding preventing software piracy, (programmers.stackexchange.com) as an example.

Though some of them are of the opinion that there is no perfect measure of totally preventing software piracy (schneier, 2001). so many others have not relented, rather they have implored a combined measure whereby the various piracy prevention measures are being combined together in order to achieve more difficult-to-pirate software. Legitimate end-users have not been left aside as well in this fight for software piracy prevention since they tend to suffer the pain of price increase when software vendors try to make up for losses incurred from pirates. Legitimate users have been continuously charged to report software piracy through electronic means in so many websites today. Issuing licenses to end-users by the software vendors at purchase of software and prompting users to register online when installing a product has also been in practise, but pirates rarely register. Technical support groups are likewise, rarely used by pirates, given their reluctance to disclose their illegal use of the software (Adu and Araoye, 2013).

These various software piracy prevention schemes have gone a long way in preventing software piracy. But unfortunately, according to these reviews, each of these measures has been seen to have their various shortcomings, owing to the fact that several end-users have their various reasons for owning and using software illegitimately even when it feels unappealing to them. An argument in favour of software piracy was raised by Solove, (2012). Solove argued that when viewed in a historical context, the benefits of software piracy far outweigh its short-term costs. He argued that Software pirates promote data survival through ubiquity and media independence giving an example of when software publishers spent countless man-hours in the 1980s preventing people from archiving their work by devising schemes to forever lock their software onto a single, authorized diskette.

One popular copy protection method involved placing an intentionally corrupt block of data on a disk to choke up error-checking copy routines. It worked so well that it also prevented honest attempts to back-up legally purchased software. He also continued that if those copy protection schemes had been foolproof, as intended, and copyright law had been obeyed, most of the programs published on those fading disks would now be gone forever. Many cultural touchstones of a generation would have become extinct due to greed over media control. Following the comments on his arguments, majority applauded him. This has shown that software piracy very much appeal to pirates, amongst other reasons of their making profits from the act. It is therefore expected of software vendors and manufacturers, as well as programmers not to relent in introducing and implementing further technical means of preventing software piracy.

This writing analyses the use of Code Encryption Techniques, The technique has a pre-stored hash value of any software it is integrated to. Every time the software is accessed, it computes a hash value and compares it with the existing pre-stored hash value. If both hash values are the same, the software runs otherwise it terminates. Apart from this feature, it also converts all characters of the serial number to hexadecimal or mashed data so it is very difficult to know the serial number needed to access the software, therefore it is another way of contributing to the relentless efforts which is aimed at the prevention of software piracy. The next chapter will introduce the Code encryption and their behaviors, how they can be used to fight software piracy, developing Code Encryption Techniques for piracy prevention, showing how to use the system and using analysis made from data collection procedures to ensure an outstanding contribution of Code Encryption Techniques to the fight against software piracy, but first ensuring that data collected are from population of software users who are ‘really informed’ or who have proper education about

REFERENCES

Cronin, G. (2002). A taxonomy of methods for software piracy prevention. *Department of Computer*

*Science, University of Auckland, New Zealand, Tech. Rep*.

Korhonen, J. (2015). *Piracy prevention methods in software business* (Doctoral dissertation, Bachelor’s

Thesis, University of Oulu-Department of Information, Oulu, Finland. Retrieved from h<ttp://jultika.> oulu. fi/files/nbnfioulu-201605131733. pdf).

Cliff, D. T., Low, C. A., & Hawkes, R. J. (2011). *U.S. Patent No. 8,042,050*. Washington, DC: U.S. Patent

and Trademark Office.

Collberg, C. S., & Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation-tools for

software protection. *IEEE Transactions on software engineering*, *28*(8), 735-746.

Cronin, G. (2002). A taxonomy of methods for software piracy prevention. *Department of Computer*

*Science, University of Auckland, New Zealand, Tech. Rep*.

Cronin, B. (2001). Hyperauthorship: A postmodern perversion or evidence of a structural shift in scholarly

communication practices?. *Journal of the American Society for Information Science and Technology*, *52*(7), 558-569.

Mykhalchenko, N., & Wiegratz, J. (2019). Anti-fraud measures in Southern Africa. *Review of African*

*Political Economy*, *46*(161), 496-514.

Gates, Kelly. "Will work for copyrights: The cultural policy of anti-piracy campaigns." *Social Semiotics* 16,

no. 1 (2006): 57-73.

Collard, R. C., Dempsey, J., & Sundberg, J. (2015). A manifesto for abundant futures. *Annals of the*

*Association of American Geographers*, *105*(2), 322-330.

Moores, T. T., Nill, A., & Rothenberger, M. A. (2009). Knowledge of software piracy as an antecedent to

reducing pirating behavior. *Journal of Computer Information Systems*, *50*(1), 82-89.

Jason-Lloyd, L. (2013). *The legal framework of police powers*. Routledge.

Orme, T. (2014). The short-and long-term effectiveness of anti-piracy laws and enforcement

actions. *Journal of Cultural Economics*, *38*(4), 351-368.

Mach, L. T., & Nash, C. (2019). Social media versus traditional Vietnamese journalism and social power

structures. *Asian Journal of Journalism and Media Studies*, *2*, 1-14.

Law, I. (2013). *Racism and ethnicity: global debates, dilemmas, directions*. Routledge.

McMullen, K. F. (2014). Worlds collide when 3D printers reach the public: Modeling a digital gun control

law after the digital millenium copyright act. *Mich. St. L. Rev.*, 187.

Litman, J. (1996). Reforming Information Law in Copyright's Image. *U. Dayton L. Rev.*, *22*, 587.

Harvey, M. (2002). What’s so special about legal translation?. *Meta: journal des traducteurs/Meta:*

*Translators' Journal*, *47*(2), 177-185.

Nichols, K. (1999). The age of software patents. *Computer*, *32*(4), 25-31.

Dakin, K. (1997). What if there were no software piracy?. *IEEE software*, *14*(1), 20-21.

Bagchi, K., Kirs, P., & Cerveny, R. (2006). Global software piracy: can economic factors alone explain the

trend?. *Communications of the ACM*, *49*(6), 70-76.

Allaire, J. (2002). Macromedia Flash MX-A next-generation rich client. *Macromedia white paper*, 1-2.

Cronin, G. (2002). A taxonomy of methods for software piracy prevention. *Department of Computer*

*Science, University of Auckland, New Zealand, Tech. Rep*.

Cronin, G. (2002). A taxonomy of methods for software piracy prevention. *Department of Computer*

*Science, University of Auckland, New Zealand, Tech. Rep*.

Stalberg, E., & Cronin, C. (2011). Assessing the cost and value of bibliographic control. *Library*

*Resources & Technical Services*, *55*(3), 124-137.

Larsson, J. E., Richardson, A., & Steckler, P. A. (2001). *U.S. Patent No. 6,226,747*. Washington, DC: U.S.

Patent and Trademark Office.

Asongu, S. A., Singh, P., & Le Roux, S. (2018). Fighting software piracy: some global conditional policy

instruments. *Journal of Business Ethics*, *152*(1), 175-189.

Yang, D., Sonmez, M., Bosworth, D., & Fryxell, G. (2009). Global software piracy: Searching for further

explanations. *Journal of Business Ethics*, *87*(2), 269.

Bahar, R. (2011). *U.S. Patent Application No. 13/196,158*.

Andrew, F. (2013). *U.S. Patent No. 8,620,667*. Washington, DC: U.S. Patent and Trademark Office.

Davidson, R. I., & Myhrvold, N. (1996). *U.S. Patent No. 5,559,884*. Washington, DC: U.S. Patent and

Trademark Office.

Cappaert, J. (2012). Code obfuscation techniques for software protection. *Katholieke Universiteit Leuven*,

1-112.

Crusoe, J. (2019). *Why is it so challenging to cultivate open government data?: Understanding*

*impediments from an ecosystem perspective* (Vol. 124). Linköping University Electronic Press.

Myles, G., & Collberg, C. (2003, November). Software watermarking through register allocation:

Implementation, analysis, and attacks. In *International Conference on Information Security and Cryptology* (pp. 274-293). Springer, Berlin, Heidelberg.

Small, D. (2010). *Subjectified: personification as a design strategy in visual communication* (Doctoral

dissertation, Massachusetts Institute of Technology).

Cronin, G. (2002). A taxonomy of methods for software piracy prevention. *Department of Computer*

*Science, University of Auckland, New Zealand, Tech. Rep*.

Djekic, P., & Loebbecke, C. (2007). Preventing application software piracy: An empirical investigation of

technical copy protections. *The Journal of Strategic Information Systems*, *16*(2), 173-186.

Olsson, J., & Luchjenbroers, J. (2013). *Forensic linguistics*. A&C Black.

Jakobsson, M., & Reiter, M. K. (2001, November). Discouraging software piracy using software aging.

In *ACM Workshop on Digital Rights Management* (pp. 1-12). Springer, Berlin, Heidelberg.

Schneier, B. (2001). Insurance and the computer industry. *Communications of the ACM*, *44*(3), 114-114.

Adu, M. K., & Araoye, K. A. A. I. Identifying Major Reasons for Software Piracy in Developing Nations.

Solove, D. J. (2012). Introduction: Privacy self-management and the consent dilemma. *Harv. L.*

*Rev.*, *126*, 1880.

# CHAPTER THREE

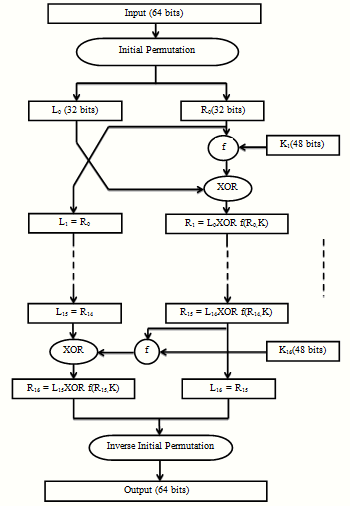
# System Analysis and Design

## 3.1 Data Encryption Algorithms

In this section, we consider the various algorithms use in data encryption and their various procedures

**3.1.1 Data Encryption Standard (DES)**

DES is a block cipher. It encrypts the data in a block of 64 bits. It produces 64 bit cipher text. The key length is 56 bits. Initially the key is consisting of 64 bits. The bit position 8, 16, 24, 32, 40, 48, 56, 64 discarded from the key length (Wunnavaand Rassi, 2002). Figure 3.1 shows the Diagram of DES.



***Figure 3.1: Diagram of DES. Source:*** (Singh *et al*, 2013)

DES is based on two fundamental attributes of cryptography: Substitution (confusion) and transposition (Diffusion). DES consists of 16 steps, each of which called as a Round.

**Algorithm:**

DES algorithm is shown below:

**Step**

1. In the first step, the initial 64-bit plain text block is handed over to Initial Permutation (IP) function.
2. The Initial permutation is performed on plain text.
3. The initial permutation produces two halves of permuted block: Left Plain text (LPT) and Right Plain (RPT).
4. Now, each of LPT and RPT goes through 16 rounds of the encryption process, each with its own key:
5. From the 56-bit key, a different 48-bit Sub-key is generated using Key Transformation.
6. Using the Expansion Permutation, the RPT is expended from 32 bits to 48 bits.
7. Now, the 48-bit key is XORed with 48-bit RPT and the resulting output is given in the next step.
8. Using the S-box substitution produce the 32-bit from 48-bit input.
9. These 32 bits are permuted using P-Box Permutation.
10. The P-Box output 32 bits are XORed with the LPT 32 bits.
11. The result of the XORed 32 bits is becomes the RPT and old RPT become the LPT. This process is called as swapping.
12. Now the RPT again given to the next round and performed the 15 more rounds.

V. After the completion of 16 rounds the Final Permutation is performed (Kahte, 2003; Mahajan, and Sachdeva, 2013).

## 3.1.2 Double DES

It is also called 2DES. Its process is the same as DES but repeated the same process 2 times using two keys K1 and K2. First it takes plain text, produced the cipher text using K1 and then take up the cipher text as input, produced another cipher text using K2 shown in figure 3.2. The Decryption Process is shown in figure 3.3 (Connell, 2009).

Cipher text

Plain text

Encrypt

With

K1

Cipher text

Encrypt

With

K2

***Figure 3.2: Encryption with key K1 and K2. Source:*** (Singh *et al*, 2013).

Plain text

Cipher text

Decrypt

With

K2

Cipher text

Decrypt

With

K1

***Figure 3.3: Decryption with key K1 and K2. Source:*** (Singh *et al*, 2013)

**3.1.3 Triple DES**

Triple DES is DES -three times. It comes in two flavors: One that uses three keys and another that uses two keys. The Idea of 3-DES is shown in the figure 3.4. The plain text block P is first encrypted with a key K1, then encrypted with second key K2, and finally with third key K3, where K1, K2 and K3 are different from each other. To decrypt the cipher text C and obtain the plain text, we need to perform the operation P= DK3 (DK2 (DK1(C))).

**Algorithm**

Triple DES with two keys, the algorithms work as follows:

**Step**

1. Encrypt the plain text with key K1. Thus, we have EK1 (p).
2. Decrypt the output of step1 above with key K2. Thus, we have DK2 (EK1 (P)).
3. Finally, encrypt the output of step 2 again with a key K1.Thus, we have EK1 (DK2 (EK1 (P))) [31]. The idea of 3-DES with two keys are shown in figure 3.5.

Encrypt

With

K3

Plain Text

Encrypt

with

K1

Cipher Text 1

Encrypt

with

K2

Cipher Text 2

Cipher Text 3

***Figure 3.4: Encryption with key K1, K2 and K3. Source:*** (Singh *et al*, 2013).

Encrypt

With

K3

Plain Text

Encrypt

with

K1

Cipher Text 1

Encrypt

with

K2

Cipher Text 2

Cipher Text 3

***Figure 3.5: Decryption with key K1 and K2. Source:*** (Singh *et al*, 2013)

**3.1.4 The RSA Algorithm**

This is a public key encryption algorithm developed by Ron Rivest, Adi Shamir and Len Adlemen in 1977.The RSA algorithm is the most popular and proven asymmetric key cryptographic algorithm. The RSA algorithm is based on the mathematical fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product. The private and public keys in the RSA are based on very large (made up of 100 or more digits) prime numbers. The algorithm itself is quite simple (unlike the symmetric key cryptographic algorithms). However, the real challenge in the case of RSA is the selection and generation of the public and private keys. The idea of asymmetric cryptographic is shown in figure 3.6. In which the A is the sender and B Receiver (Adleman, 2009).

Plain text

Sender (A)

Plain text

Cipher text

Encrypt

with B’s public key

Cipher Text

Receiver (B)

Decrypt

with B’s private key

Internet

***Figure 3.6: Decryption with Key K1 and K2. Source:*** (Singh *et al*, 2013)

**Algorithm:**

RSA algorithm is presented below:

**Step**

1. Choose two large prime numbers P and Q.
2. Calculate N= P \* Q.
3. Select the public key (i.e. the encryption key) E such that it is not a factor of (P-1) and (Q-1).
4. Select the private key (i.e. the decryption key) D such that the following equation
5. is true: (D \*E) mod (P-1) \* (Q-1) =1
6. For encryption, calculate the cipher text CT from text PT as follows: CT = PTE mod N (Adleman,2009; Kahte, 2003).

## 3.2 Research Design

The methodology that will be use in this research is the Objects-Oriented Analysis (OOA). The object oriented analysis looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed. The result of object-oriented analysis is a description of what the system is functionally required to do in the form of a conceptual model that will typically be presented as a set of use cases. The use case diagram below (see figure 3.7) is the use case boundary diagram of the system.



**Note:** In the above boundary diagram, the large rectangle is the system boundary. Everything inside the rectangle is part of the system under development. Outside the rectangle are the actors that act upon the system.

Actors are entities outside the system that provide the stimuli for the system. Typically, they are human users, or other systems. Inside the boundary rectangle are the use cases. These are the ovals with names inside. The lines connect the actors to the use cases that they act upon.

An <<includes>> relationship indicates that the second use case is always invoked by the first use case.

An <<extends>> relationship indicates that the second use case may optionally invoke the first use case.

## 3.2.1 Object-Oriented Design of the System

Object-Oriented Design (OOD) transforms the conceptual model produced in object-oriented analysis to take account of the constraints imposed by the chosen architecture and any non- functional technological or environmental-constraints, such as transaction throughput, response time, run-time platform, development environment, or programming language. The concepts in the analysis model are mapped onto implementation classes and interfaces. The result is a model of the solution domain, a detailed description of how the system is to be built. Thus, class diagram, sequence diagrams, and deployment diagrams of Unified Modeling Language (UML) will be used for the design of the system.

## Class Diagram of the System

UML class diagrams allow us to denote the static contents of, and relationship between classes. A class is depicted on the class diagram as a rectangle with three horizontal sections. The upper section shows the class name (such as User, DES, RSA, etc), the middle section contains the class attributes, and the lower section contains the class methods or operations.

The figure 3.8 shows the class diagrams of the system

Code encryption

- encrpt: Cipher

- decrypt Cipher

- initializationVector : byte[ ]

+ encrypt ()

+ decrypt ()

+ writeBytes ()

FormMain

- fileMenu: JMenu

- menuBar: JMenuBar

- saveToFile: JemnuItem

- loadFromFile: JmenuItem

+ initComponents ()

+ loadFromFile()

+ saveToFile()

+ writeLog(text:string)

+ desEncyptActionPerformed()

+ rsaEncryptActionPerformed()

+ desDecryptActionPerformed()

+rsaDecryptActionPerformed()

Code decryption

- generator:KeyPairGenerator

- encryptCipher Cipher

- decryptCipher: Cipher

+ generateKey ()

+ wirteKey ()

+encrpt()

+ readPublicKey ()

+ decrypt ()

+ readPrivateKey()

keyPairGenerator

Cipher

Cipher

**Description**

* Rectangle represents classes, and arrows represent associations in which one object holds a reference to and invokes methods upon the other.
* A dash (-) character in front of the variables in the class icon denotes private.
* A plus (+) character in front of functions or operations in the class icon denotes public.
* The type of a variable or a function argument is shown after the colon following the variable or argument name. Similarly, the return value of a function is shown after the colon following the function.

## 3.2.3 Sequence Diagram of the System

Sequence diagrams are used to show or describe the detail implementation of system behaviors shown in the use case diagram. It also describes how a particular class’s method is implemented. Below are the sequence diagrams of some of the behaviors shown in the use boundary case diagram.

User

:FileEncryptionWindow

:FormMain

:DES

encryptFileUsing DES

desEncryptActionPerformed()

encryptedtextfile

displays the encrypted text file

encrpt()

encrypted text file

plain text file

plain text file

greenerKey ()

writeKey()

readPrivateKey()

***Figure 3.9: Encrypt file using DES sequence diagram***

**Description**

* The dashed lines hanging down from the object and the actor are called life lines. A message being sent from the object to another is shown as arrow between the life lines. Each message is labeled with a name.
* Arguments appear either in the parenthesis that follow the name or next to data tokens (the little arrows with the circles on the end).
* Time is in the vertical dimension, so the lower a message appears, the later it is sent.
* The skinny little rectangle on the lifeline of the DESFileEncryption Window object is called activation. Activations represent the time that a method executes. In this case, it shows how long the encryptUsingDes method runs.
* Messages leaving the activation to the right were sent by the encryptUsingDes method. The unlabeled arrow shows the encryptUsingDes method returning to the actor and passing back a return value.

User

:FileDecryptionWindow

:FormMain

:DES

decryptFileUsing DES

desDecryptActionPerformed()

plaintextfile

displays the decrypted text file

decrept()

plaintextfile

encrypted text file

plain text file

readKey ()

User

:RSAFileEncryptionWindow

:FormMain

:RSA

encryptFileUsing RSA

rsaEncryptActionFerformed()

encryptedtextfile

displays the encrypted text file

encrpt()

encrypted text file

plain text file

plain text file

greenerKey ()

writeKey()

readPublicKey()

**Figure 3.12: Decrypt file using RSA sequence diagram.**

## Software Architecture

The system has three basic layers: the user interface layer, logic layer and the report layer. The user interface is the layer that contains the buttons that enable the user to load, encrypt, and decrypt a text file. The logic layer implements the algorithms for encryption and decryption of text files, while the report layer gives information on the size of file before and after encryption.

USER

DES ENCRYPTOR

RSA ENCRYPTOR

DES DECRYPTOR

RSA DECRYPTOR

LOGIC LAYER

REPORT LAYER

ENCRYPTED SOFTWARE

DECREPTED SOFTWARE

GENERATE REPORT

LOAD TEXT FILE

ENCRYPT USING DES

ENCRYPT USING RSA

DECRYPT USING DES

DECRYPT USING RSA

USER INTERFACE

**Figure 3.13: Architecture of the system**

The techniques use in this study or research design has features similar to the activation code system in terms of authentication and serial key features, but operates in a way that surpasses the limitations of an activation code system in terms of piracy prevention. Unlike in the activation code system earlier explained, where a user successfully installs a software product but will need a key to unlock its features for use, this Code Encryption Technique is designed such that the system requests for the key even before the installation of the software. Also, the Code Encryption Technique does not ask the user to enter his or her personal information, like in the activation code system where the user can provide false information. It rather migrates into the users’ computer and collects unique information like the Hard-disk information, Processor ID, Computer name, etc. This is done without the knowledge of the user, and thus because of the uniqueness of the Processor ID of every computer, a particular software associated with its key will be hardtobeinstalledillegally.Inotherwords,theactivationcoderequest for information from the user, whereas this Code Encryption Technique automatically collects all the information it needs making it impossible for the user to enter false information. Also, the activation code system does not store unique information, but the Code Encryption Technique stores unique information of all users mapped to their product serial keys and stores them in a database for routine check whenever different users wants to install any of the vendor’s software. This work describes a scenario where the software only has to communicate between the user’s serial key and a software key for the purpose of authentication. During software installation, the serial key of the software will migrate from the user’s own to the original software to authenticate the genuineness of the software, to ensure that the user is not trying to pirate the particular software product or to ensure that particular software product has not exceeded its maximum installation/usage limit. In other words, the vendor can limit the number of times an original purchaser can install genuine software as well as a third party userThe encryption technique which is also referred to as the hashing mechanism has a pre-stored hashed value of any software it is integrated to. Every time the software is accessed, it computes a hash value and compares it with the existing pre-stored hash value. If both hash values are the same, the software runs if not it terminates. Apart from this feature, it also converts all characters of the serial number to hexadecimal mashed data so it is very difficult to know the serial number needed to access the software. figure 1 shows an architecture of the encryption mechanism in which a plain program that is easy to understand after development by the developers is converted to encrypted form which is hard to understand and can only be decoded by someone having the decryption key.

**3.4 Key Challenges Involved In Combating Software Piracy**

In adequate reinforcement of the law

Collectivism behavior

Novelty seeking among user

Believe that software piracy act as a means of transferring technology

Inadequate knowledge on the consequences of software piracy

**3.4.1 INADEQUATE REINFORCEMENT OF THE LAW**

Inappropriate legislation of the law and its reinforcement has hampered with efforts to prevent software piracy. Many software users in many developing countries are not aware of the legal implications of software piracy (Theo Papadopoulos, 2004) and that they will see nothing wrong whether they use the original or the pirated versions of the software. As Eric kin wai Lau puts it, many software users have little or no knowledge at all of copyright law or the consequences of software piracy. They will always engage in software piracy so long as everybody else is doing it.

**3.4.2 COLLECTIVISM BEHAVIOR**

Many cultures in developing countries are more groups oriented and this leads to the violation of intellectual property rights.

According to Fang Wang *et al* (2005), software users who are group oriented have a common attitude of sharing what they have. This means that they are more likely to share computer software leading to increased piracy. Their belief is that if you visit my house and go with a copy of my software, you are doing so with my permission and thus you are committing no offence. In such other cultures, consumers are informationally susceptible where they buy software based on the opinions of others. They depend on other peoples’ decisions on the quality of particular software. Others are normative susceptible where they make purchase decisions so as to impress others (Swee Hoon Ang *et al,* 2001). In all these situations, the behavior of consumers is influenced by others and this leads to group shopping increasing purchases of pirated software. According to Md.madbubur Rahim *et al,* (2000). collectivism behavior is also common among academicians where they share copies of pirated software.

**3.4.3 NOVELTY SEEKING AMONG USERS**

This is personal gratification where the consumer will use pirated software to fulfill his/her own self-interest. According to Swee. Hoon Ang *et al,* (2001). novelty seeking concerns an orientation towards self accomplishment, recognition by other people and enjoying good and finer things in life. Consequently, this will boost a consumers’ attitude towards pirated software to achieve his/her interest. Software piracy is just a means of trying and testing new software and especially if computer literacy levels are low in a particular country (Fang Wang, *et al,* 2005). Belief that Software piracy act as transfer of knowledge Most of the developing countries view software piracy as a means of transferring technology. According to Eric Kin- Wai Lau, (2006). they argue that developed countries are selfish in technological advancement and that they prevent technological development in those countries in the name of what they call protection of intellectual property rights. According to Gael McDonald and Christopher Roberts (1994), these countries view software piracy as a roadmap to economic development and homegrown innovation. They cite examples of Hong Kong, Japan, Korea, Singapore and Taiwan which have resolved to invitation of piracy to economic growth and some of which have managed so far after the world war.

**3.4.4 BELIEVE THAT SOFTWARE PIRACY ACT AS A MEANS OF TRANSFERRING TECHNOLOGY**

Belief that Software piracy act as transfer of knowledge Most of the developing countries view software piracy as a means of transferring technology. According to Eric Kin- Wai Lau (2006). they argue that developed countries are selfish in technological advancement and that they prevent technological development in those countries in the name of what they call protection of intellectual property rights. According to Gael McDonald and Christopher Roberts, (1994). these countries view software piracy as a roadmap to economic development and homegrown innovation. They cite examples of Hong Kong, Japan, Korea, Singapore and Taiwan which have resolved to invitation of piracy to economic growth and some of which have managed so far after the world war.

**3.4.5 INADEQUATE KNOWLEDGE ON THE CONSEQUENCES OF SOFTWARE PIRACY**

In developed countries, there is no such a thing like educating a consumer who well knows that piracy is not allowed (Fang Wang, et al, 2005). However in many developing countries and given that there is newly expanding market growth, an education process is required to inform the consumers that it is wrong to pirate as the majority have no such knowledge. If the consumers do not know that the pirate’s customer is as guilty as the pirate, they are unlikely to cease from software piracy than when they are aware of the consequences involved when caught in the act. Eric Kin -Wai Lau, (2007.) adds that consumer users of software in most developing countries have no knowledge of the copyright law and this leads to leniency towards software piracy. They would not mind buying pirated software as long as everybody else within the society is doing it. Inadequacy of knowledge on the consequences of software piracy contributes positively to the leniency towards software piracy. A case is given in Jordan where the penalty for violating copyright law was set at a sentence for not less than three months and a fine of not less than JD 1000 to organizations and consumers violating copyright (Saleh Al Sharari, 2006). after which there were reported reduction in software piracy.

**3.5 Method of data Collection**

The study adopts the survey research method, which is appropriate in assessment studies as this. This method selects and studies samples drawn from a population to discover the relative incidence, distribution and inter-relations of sociological and professional variables. Since this study is an audience research necessitating opinions, attitudes, motivations and individualistic consideration, the survey method was considered most appropriate.

# 3.6 Population of the Study

Population “typically is a well defined set of people”. It refers to a total number of people living in a particular place or performing similar activity. Therefore, the target populations for this study are professionals in the software industry, undergraduate and postgraduate students of Kebbi State University of Science and technology. The population required to ensure that this Combating Software Piracy using Code Encryption Technique has an outstanding contribution to make to the fight against software piracy are those who are ‘really informed’ about software piracy.

# 3.7 Sample Size and Technique

A total of 150 software professionals will be surveyed in order to achieve the research aim and objectives. Respondents from the software development industry will be randomly surveyed. Hence, purposive random sampling technique will be adapted to survey professionals from the software industry in order to give concise and accurate information to achieve the research aim and objectives.

3.8 **Instrument for Data Collection**

The researcher designed an Electronic mean as the data collection instrument for this study. The questionnaires were administered solely to professional software users. The questions were aimed at eliciting relevant information of software Piracy in Birnin-Kebbi. Questions relating to methodology and material for software use, perceived problems, as well as possible strategies that could be adopted to enhance software use in Birnin-Kebbi were asked in the electronic questionnaire.

A questionnaire designed by the researcher titled “Combating software piracy using code encryption” was also used in the study. The content of the instrument was based on the findings of the questionnaire conducted with the software users in various places Birnin-Kebbi as well as on the information from the literature reviewed.

The questionnaire has five sections: A, B, C, D and E:

Section “A”, is on personal data of the respondents;

Section “B”, is on the needs/objectives of Software;

Section “C” contains questions on the perceived problems of Software use.

Section “D”, on the income involved in purchasing software in Birnin-Kebbi.

Finally, section “E”, made up of the possible strategies that could be adopted for improved software use in Birnin-Kebbi.

The instrument was structured in the *modified Likert* fashion, on a 4 – point scale,

Ranging from “strongly agree” (SA), through “agree” (A), “disagree” (D) to “strongly

Disagree” (SD). Subjects were then instructed to respond to their degree of agreement with the statements contained in the instrument.

# 3.9 Validity and Reliability of the Research Instrument

The questionnaire, being part of the instrument used in conducting this research, will be fashioned in such a way as to adequately reflect the phenomenon under study. The questions in the questionnaire will be peer reviewed to ensure that right and relevant questions are asked. The questions will be reviewed to eliminate non-clarity and ambiguity. Moreover, the questionnaire will be drafted and approved under the guidance of the project supervisor. It will be ensured that the instrument has due assent before distribution.

# 3.10 Method of Data Collection

After the pilot testing and all necessary modifications, the questionnaires were administered, the questionnaires was distributed and retrieved online via various online media i.e. WhatsApp, E-mail, etc. to the chosen sample for the study. Fifty copies of the questionnaire given out were successfully completed and returned. The possibility of retrieving back all the questionnaire was as a result of Microsoft Google online tool used to design questionnaires due to the fact that its easy, fast and can cover a wide range of respondents.

# 3.11 Method of Data Presentation and Analysis

Having gathered the data through the administration of questionnaires, the collected data will be coded, tabulated, and analyzed according to the research questions and objectives.

The data’s will be analyzed statistically using SPSS (Statistical Package for Social Sciences) version 23. In the analysis of the data, a 95% confidence level will be adopted and values less than 0.05 will be considered to be statistically significant while values that are greater than 0.05 will be considered not statistically significant.

Data presentation will also be done using tables and histograms in order to analyze the data collected effectively and efficiently for easy management and accuracy, the simple percentage method, likert scale, and Chi-square test will be the major analytical tools used for this research project with any other analytical tool that might be necessary during the analysis stage

3.11.1 **Likert Scale Analysis**

The likert scale analysis will be done using the Relative Importance Index (RII) to rank responses of various questions from the questionnaire. The Relative Importance Index (RII) is given as;

Relative Importance Index (RII) =

Where = ∑*fx*= is the weight given to each factor by the respondents and ranges from 1 to 5,

|  |  |  |  |
| --- | --- | --- | --- |
| Strongly Agree(SA) | Agree(A) | Disagree(D) | Strongly Disagree(SD) |
| 4 Points | 3 Points | 2 points | 1 Point |

# ∑*f =* is the total number or respondents in the sample

# K = is the highest weight on the likert scale (i.e. 4 = strongly agree)

# Ranking of the items under consideration was based on their RII values. The item with the highest RII value is ranked first (1) the next (2) and so on.

# 3.11.2 Chi- Square Specification

The Chi-Square test is used to determine if a sample comes from a population with a specific distribution. This test is applied to binned data, so the value of the test statistic depends on how the data is binned. The test statistic, which is a random variable, is a function of the observed and expected frequencies, which are also random variables**.** It is also refers to as goodness of its test. The formula for chi-square method is sated as:

X2 = (O – E) 2

E

Where:

O = observed distribution

E = expected distribution

∑ = summation symbol

X2 = chi-square

# 3.11.2.1 Justification of the Tools of Analysis

Chi-square statistics is the main tool of analysis adopted for the study because chi-square is a non-parametric measure that is given to analyze data that are organized in the form of frequency counts rather than measuring magnitude. Chi-square is normally used statistically to show the goodness or fit. It is adopted for this study because it is by nature non-negative (i.e. X2>0) and is used to weigh the relative importance in the view of different categories of people on sensitive issues or matters of public importance.

**CHAPTER FOUR**

**PRESENTATION AND ANALYSIS OF DATA**

* 1. **INTRODUCTION**

This chapter contains presentation of a collection from the field work which was analysis using statistics package for social science (SPSS) version 20 and interpreted according. The results of the data analysis are also discussed in the chapter

**4.2. PRESENTATION AND INTERPRETATION OF DATA**

Electronic Questionnaires ranging from 1-150 were administered to the software professionals and Students of Kebbi state University of Science and Technology Aliero, through electronic media such as E-mail, WhatsApp, etc. However, 148 questionnaires were dully completed and returned electronically, representing a response of 98% and a non-response of 2%. 2 questionnaires were not returned by some of the Student of the selected software professionals for one reason or the other.

Below is a hypothetical tabulation of the analysis of the Electronic questionnaires that were administered.

Table 4.1.1 shows the response of respondents on the effects of Software Encryption on software Piracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Encrypted Software has effect on software Piracy | Non-Encrypted Software has effect on Software Piracy | Both Encrypted and Non-Encrypted Software has no effect on Piracy |  |
| Yes | 32(21.6%) | 24(16.22%) | 28(18.92%) | 84 |
| No | 20(13.51%) | 22(14.86%) | 22(14.46%) | 64 |
| Total | 52 | 46 | 50 | 148 |

Source: Electronic questionnaire Survey

From the table above, 32 persons representing 21.62% of the total respondents agreed that Developer encryption of software has effect on software Piracy ,24 persons representing 16.22% of the total respondents Agreed that Non-Developer encryption of software has effect on software, 28 persons each representing 18.92% of the total respondents agreed that both encryption and non-Encryption of software has effect on software Piracy, and 20 persons representing 13.51% of the total respondent Strongly disagreed that Developer encryption of software has effect on software Piracy, and 22 persons representing 14.86% of the total respondent disagreed that Non-Developer encryption of software has effect on software Piracy the same Number of respondent also disagreed that both Developer encryption and Non-Encryption has effect on Software Piracy. It can be concluded that majority of the respondents agreed that Developer encryption has significant effect on Software Piracy.

Table 4.1.2 Unemployment has no effect on the growth of Piracy in Nigeria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Unemployment has great effect of Software Piracy | Unemployment has no effect of Software Piracy | Unemployment has Nothing to do with Piracy |  |
| Yes | 70(47.29%) | 10(6.76%) | 19(12.84%) | 99 |
| No | 12(8.11%) | 16(10.81%) | 21(14.19%) | 49 |
| Total | 82 | 26 | 40 | 148 |

Source: Electronic questionnaire survey

From the table above, 70 persons representing 47.29% of the total respondents Agreed with the view that Unemployment has effect on the growth of Piracy in Nigeria. And 10 persons representing 6.76% of the total respondents simply agreed with the view that Unemployment has no effect on the growth of Piracy in Nigeria. 19 persons representing 12.84% of the total respondents agreed with the view that Unemployment has nothing to do with growth of Piracy in Nigeria. And 12 persons representing 8.11% of the total respondents Disagreed with the view that Unemployment has effect on the growth of Piracy in Nigeria. And 16 persons representing 14.86% of the total respondents Disagreed with the view that Unemployment has no effect on the growth of Piracy in Nigeria. And 21 persons representing 14.19% of the total respondents Disagreed with the view that Unemployment has nothing to do with the growth of Piracy in Nigeria It can be concluded therefore that most of the respondents were of the view Unemployment contributed hugely to piracy in Nigeria.

Table 4.1.3 Price of software has great effect on the choice of Piracy in Nigeria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Price of software has effect on software piracy | Price of software has no effect on software piracy | Price has nothing to do with Piracy |  |
| Yes | 52(35.14%) | 14(9.46%) | 12(8.11%) | 78 |
| No | 12(8.11%) | 39(26.35%) | 19(12.84%) | 70 |
| Total | 64 | 53 | 31 | 148 |

Source: Electronic questionnaire survey

From the table above, 52 persons representing 35.14% of the total respondents Agreed that the Price of software has great effect on the choice of Piracy in Nigeria. And 14 persons representing 9.46% of the total respondents Agreed that the Price of software has no effect on the choice of Piracy in Nigeria. . And 14 persons representing 9.46% of the total respondents Agreed that the Price of software has nothing to do with the choice of Piracy in Nigeria. 12 persons representing 8.11% of the total Respondents Disagreed that the Price of software has great effect on the choice of Piracy in Nigeria. And 39 persons representing 26.35% of the total respondents disagreed that the Price of software has no effect on the choice of Piracy in Nigeria. And While 19 persons representing 12.84% of the total respondents Disagreed that the Price of software has nothing to do with the choice of Piracy in Nigeria. It therefore can be concluded that most of the respondents posited that Price of software has great effect on the choice of Piracy in Nigeria.

Table 4.1.4 Government Policies has effect on the growth of Piracy in Nigeria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Government Policy has effect on the growth of software piracy in Nigeria | Government Policy has no effect on the growth of software piracy in Nigeria | Government Policy has nothing to do with Piracy in Nigeria. |  |
| Yes | 66(44.59%) | 13(8.78%) | 10(6.76%) | 89 |
| No | 12(8.11%) | 29(19.59%) | 18(12.16%) | 59 |
| Total | 78 | 42 | 28 | 148 |

Source: Electronic questionnaire survey

From the table above, 66 persons representing 44.59% of the total respondents Agreed with the view that Government Policies effect the growth of Piracy in Nigeria. And 13 persons representing 8.78% of the total respondents Agreed with the view that Government Policies on Piracy has no effect on the growth of Piracy in Nigeria. and 10 persons representing 6.76% of the total respondents agreed with the view that Government Policies on Piracy has nothing to do with the growth of Piracy in Nigeria. and 12 persons representing 8.11% of the Respondents total Disagreed with the view that Government Policies on Piracy has effect on the growth of Piracy in Nigeria. and 29 persons representing 19.59% of the Respondents Disagreed with the view that Government Policies on Piracy has no effect on the growth of Piracy in Nigeria, While 18 persons representing 12.16% of the Respondents Disagreed with the view that Government Policies on Piracy has nothing to do with the growth of Piracy in Nigeria It can be concluded that a large number of the respondents oppose the view that Government Policies on Piracy has no effect on the growth of Piracy in Nigeria.

**4.2 TEST OF HYPOTHESIS**

**HYPOTHESIS 1**

ALTERNATE (Ha): Software Encryption has effect on software Piracy.

NULL (Ho): Software encryption has no effect on Software piracy.

Response on the effect on software encryption on Software piracy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Response | Developer encryption has effect on software Piracy | | Developer Encryption has no effect on Software Piracy | | Both Encryption and non encryption has nothing to do with software Piracy | | **X2** |
|  | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies |
| Yes | 32 | 19.94 | 24 | 17.64 | 28 | 19.18 | 13.64 |
| No | 20 | 15.19 | 22 | 13.44 | 22 | 14.61 | 10.71 |
|  | | | | | | | **24.35** |

Level of significance= 0.05

Degree of freedom

(r-1) (n-1)

(2-1) (3-1)

1\*2

=2

At 0.05 level of significance, given the above degree of freedom,

Table value of X2 (i.e. X2t) =5.99

To test our hypothesis, the decision rule is

Accept Ho if X2t > X2 calculated,

And

Reject Ho if X2t < X2 calculated.

Thus since the X2 table (5.99) < X2 calculated (24.35), we reject Ho and accordingly accept Ha. We conclude by accepting the alternate hypothesis. This implies that Software encryption has effect on software Piracy.

**HYPOTHESIS II**

ALTERNATE (Ha): Unemployment has effect on the growth of piracy in Nigeria.

NULL (Ho): Unemployment has no effect on the growth of piracy in Nigeria.

Response on the effect of Unemployment on Software Piracy in Nigeria.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Response | Unemployment has effect of the growth of software piracy | | Unemployment has no effect on the growth of software piracy | | Unemployment has nothing to do with software Piracy | | **X2** |
|  | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies |
| Yes | 70 | 37.06 | 10 | 11.75 | 19 | 18.08 | 29.54 |
| No | 12 | 18.34 | 16 | 5.82 | 21 | 8.95 | 36.22 |
|  | | | | | | | **65.76** |

Level of significance= 0.05

Degree of freedom

(r-1) (n-1)

(2-1) (3-1)

1\*2

=2

At 0.05 level of significance, given the above degree of freedom,

Table value of X2 (i.e. X2t) =5.99

To test our hypothesis, the decision rule is

Accept Ho if X2t > X2 calculated,

And

Reject Ho if X2t < X2 calculated.

Thus since the X2 table (5.99) < X2 calculated (65.76), we reject Ho and accordingly accept Ha. We conclude by accepting the alternate hypothesis. This implies that unemployment has effect on the growth of software piracy in Nigeria.

**HYPOTHESIS III**

ALTERNATE HYPOTHESIS (Ha): Price of software has effect on the growth of software piracy .

NULL HYPOTHESIS (Ho): Price of software has no effect on the growth of software piracy.

Response on the effect price of software on the growth of software piracy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Response | Price of software has effect on the growth of software piracy | | Price of software has no effect on the growth of software piracy | | Price of Software has nothing to do with software piracy. | | **X2** |
|  | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies |
| Yes | 52 | 22.79 | 14 | 18.87 | 12 | 11.04 | 38.78 |
| No | 12 | 20.45 | 39 | 16.93 | 19 | 9.91 | 40.6 |
|  | | | | | | | **79.38** |

Level of significance= 0.05

Degree of freedom

(r-1) (n-1)

(2-1) (3-1)

1\*2

=2

At 0.05 level of significance, given the above degree of freedom,

Table value of X2 (i.e. X2t) =5.99

To test our hypothesis, the decision rule is

Accept Ho if X2t > X2 calculated,

And

Reject Ho if X2t < X2 calculated.

Thus since the X2 table (5.99) < X2 calculated (79.38), we reject Ho and accordingly accept Ha. We conclude by accepting the alternate hypothesis. This implies that the Price of software has effect on the growth of software piracy in Nigeria.

**HYPOTHESIS IV**

ALTERNATE HYPOTHESIS (Ha): Government policy has effect on the growth of software Piracy .

NULL HYPOTHESIS (Ho): Government policy has no effect on the growth of software Piracy.

Response on the effect Government policy on the growth of software piracy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Response | Government Policy has effect on the growth of software piracy | | Government Policy has no effect on the growth of software piracy | | Government Policy has nothing to do with software piracy. | | **X2** |
|  | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies | Observed frequencies | Expected frequencies |
| Yes | 66 | 31.69 | 13 | 17.07 | 10 | 11.38 | 38.29 |
| No | 12 | 21.01 | 29 | 11.31 | 18 | 7.54 | 46.04 |
|  | | | | | | | **84.33** |

Level of significance= 0.05

Degree of freedom

(r-1) (n-1)

(2-1) (3-1)

1\*2

=2

At 0.05 level of significance, given the above degree of freedom,

Table value of X2 (i.e. X2t) =5.99

To test our hypothesis, the decision rule is

Accept Ho if X2t > X2 calculated,

And

Reject Ho if X2t < X2 calculated.

Thus since the X2 table (5.99) < X2 calculated (84.33), we reject Ho and accordingly accept Ha. We conclude by accepting the alternate hypothesis. This implies that Government Policies has effect on software Piracy.

**4.4. SUMMARY OF FINDINGS**

The following findings were made by the researcher at the end of the study.

* Lack of developer encryption contributes to the growth of piracy in Kebbi State.
* The various policy and programmes designed by the government to curb Piracy in the country has no effect on the growth of piracy.
* Unemployment contributes majorly to the growth of software piracy in the country.
* The Price of Software in the country contribute majorly to the growth of software Piracy .

**CHAPTER FIVE**

**SUMMARY, RECOMMENDATION AND CONCLUSION**

**5.1. INTRODUCTION**

This chapter consists of the summary of findings produced by the study concerning Combating software piracy in Nigeria using code Encryption, followed was conclusion and recommendations to the study. On the basis of the analysis of data collected during the electronic interview, the researchers present the following findings in order of ranking

**SUMMARY**

The following findings were made by the researcher at the end of the study

* The findings shows that Developer encryptions has great effect and can influence the rate of software piracy In Nigeria, though some are still of the opinion that both developer encryption and non-encryption of software has no effect on software piracy.
* The various Policies and measures put in place by the government to curb software piracy in the country has yielded nothing, Many believe government policies on software piracy are to light and has no effect on software piracy in Nigeria.
* It was also deducted that absence of Job opportunities (Unemployment) serves a major effect that leads to software piracy in the country.
* The finding also shows that the Price of software contributes greatly to the software piracy in the country as many cannot afford to pay for the exorbitant price of software from a developer source.

**5.2 RECCOMMENDATIONS**

in the light of the studied questionnaire obtained electronically from software professionals through different electronics medias, the researcher was able to make the following recommendations.

* REGULATION OF GOVERNMENT POLICY ON SOFTWARE PIRACY.

In the course of the study, the researcher was able to gather that government policies, punishment and law on software piracy were to light and hence encourages many to get involved in software piracy. In view of this, the researcher recommends that government should make policies that will apprehend piracy immediately and make serious punishment as a law to those that are found guilty of the crime, Government school also device a means to scrutinize the internet for effective apprehension of hackers and bring them to Justice.

* PRICE OF SOFTWARE

The price of software from developer source was also found to be too costly for the software user, which in turn result to software piracy, The researcher hence recommends that the price of software be dropped to a considerable amount that is affordable by many, this will discourage piracy and encourage developers and distributors as many will patronize genuine software instead of pirated once.

* SOFTWARE ENCRYPTION.

During the course of the study, the researcher gathered that developer encryption of software contribute greatly to the reason behind software piracy, he therefore recommends that Developers and distributors of software should encrypt their software for better protection and discouragement of software piracy.

* UNEMPLOMENT

Non availabilities of job opportunities have also led many to software piracy as many of them sees piracy as a means of making money and earn a living, Therefore the researcher recommends that government should provide job opportunities for the youth and the vulnerable ones to earn a living as this will discourage them from hacking, stealing, and pirating softwares.

**5.3 CONCLUSION**

Software piracy and the breach of the copyright laws either intentionally or unintentionally which is quite common these days. Software piracy is a menace to software developers and computer users in Nigeria all over the world. Software hackers have become nuisance to many organizations, corporate bodies and government alike. Pirating software has caused lost of several billions US Dollars in revenue and the problem continued unabated. There have been a lot of security threats in recent past due to the activities of hackers. Several financial organizations and national securities have been threatened and even some have been compromised. In this paper, we proposed the code encryption technique for combating software piracy. The technique converts plain code to an encrypted for that cannot be understood by the hacker or intended hacker unless he has the key to encrypt or decode the encrypted data, Job opportunities should also me provide to discourage this activities, The price of software should also be lowered and above all, government policies should be made more intense and real as to bring those found guilty into justice, The aforementioned factors if well tackled will go a long way to discourage software piracy in Nigeria.