Malware Propagation, Encryption and Re-randomization

By

Ahsan Rasheed Abbasi

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Science and Technology, Rawalpindi in partial fulfilment of the requirements for the degree of MS in Information Security

November 2019

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS thesis written by Mr. **Ahsan Rasheed Abbasi**, Registration No. **00000171584**, of **Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfilment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and the local evaluators of the scholar have also been incorporated in the said thesis.

Signature: _____

Name of Supervisor: **Dr Mehreen Afzal**

Date: _____

Signature (HOD): _____

Date: _____

Signature (Dean): _____

Date: _____

Declaration

I, *Ahsan Rasheed Abbasi* declare that this thesis titled "Malware Propagation, Encryption and Re-randomization" and the work presented in it are my own and has been generated by me as a result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a Master of Science degree at NUST
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated
3. Where I have consulted the published work of others, this is always clearly attributed
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work
5. I have acknowledged all main sources of help
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Ahsan Rasheed Abbasi,
00000171584

Copyright Notice

This thesis is dedicated to *my beloved parents* and *teachers* who supported me each step of the way.

Acknowledgments

All praises to Almighty Allah for giving me the strength, knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Without His blessings, this achievement would not have been possible.

I would like to convey my gratitude to my supervisor, Dr Mehreen Afzal, for his time, generous guidance, patience and encouragement. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works are major contributions to the success of this research. Also, I would thank my committee members; Dr Muhammad Faisal Amjad, and Dr Fawad khan for their support and knowledge regarding this topic.

Last, but not the least, I am highly thankful to my parents. They have always stood by my dreams and aspirations and have been a great source of inspiration for me. I would like to thank them for all their care, love and support through my times of stress and excitement.

Abstract

The dramatic growth in encrypted traffic changes the security landscape. As more organizations are becoming conscious of the need to protect their data, more devices, services and applications use encryption as the fundamental way to secure information. As organizations progressively use encryption to keep confidential their network information, attackers use the technology to camouflage their activities. In other words, encryption, which is essential for the protection of sensitive information such as online transactions, e-mails and smartphone applications, can make it possible for malware that hides within that encrypted traffic to move through the security system of an organization uninspected. Encrypting malware payload prevent malware analyst, to reverse engineering of malicious code and identifying malware developer's intension. This thesis evaluates malware encryption scheme based on ElGmal cryptosystem as a proof of concept. The thesis also present the novel scheme for malware encryption, propagation and re-randomization using environmental keys, based on Paillier cryptosystem. Furthermore, the thesis includes the review of existing methods for encrypted malware traffic analysis.

Keywords: *malicious cryptography, homomorphic encryption, re-encryption, ElGamal, Paillier cryptosystem*

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

HE	Homomorphic Encryption
PHE	Partially Homomorphic Encryption
FHE	Fully Homomorphic Encryption
GCD	Greatest Common Divisor
LCM	Least Common Multiple
RAT	Remote Access Trojan
RSA	Rivest, Shamir, and Adelman
IBE	Identity Based Encryption
BSS	Blind Source Separation
MAC	Media Access Control
ELF	Executable and Linkable Format
OAEP	Optimal Asymmetric Encryption Padding
SSL	Secure Sockets Layer
TLS	Transport Layer Security
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator

Chapter 1

Introduction

1.1 Introduction

Malware is a malicious software which is used or developed by attackers to achieve their intended objectives by running on target computer system without client authorization The purpose of the malware author is to breach the confidentiality, integrity or availability of the target system to access private computer networks, collect sensitive information and disrupt the normal computer operations.

Early information systems implementation was far more vulnerable, however malicious threats weren't a issue. It was just because quite few individuals were able to comprehend profoundly how information systems operate and therefore exploit them during those early years. Many early infectious programs have been developed as demonstrations or pranks.

Computers have become increasingly popular with technological advancements and the internet has been evolved as an increasingly important part of everyday life. The best known malware categories are recognized not because of any specific types of behaviours, but because of the way they spread. Most popular malicious software categories, including viruses, worms, trojans, spyware, adware and hijackers. The advancement and spread of all types of malware includes viruses, warms and trojans for every individual user is becoming a real concern. As personal computers, smartphones and other smart devices are constantly connected internet, the problem becomes increasingly more difficult. Potential victims are more than ever before for malicious threats. Today, malware is mainly used to obtain useful domestic, financial, or organizational data for the ben-

efit of others. Malware is sometimes widely used against state or corporate websites to retrieve confidential data or interfere with its operation in particular. However, malware is often used against individuals to obtain sensitive data such as financial security numbers, bank or payment card details, and so on. The extensive use of networks and the Internet raises the potential of this type of software to spread effectively.

Malware targets applications and services running over the internet. As almost all the organizations and firms use the internet to improve their service quality, the need to detect and deactivate malware as soon as possible so that the adverse effects created by these malware can be avoided. Malware which has the capacity to propagate is most harmful because there is no central control, so defending them is not a trivial task. Malware authors always come with new ideas. They develop malicious software in such a way that they have altered themselves so that they can not be easily detected.

1.1.1 Malicious use of Cryptography

Cryptography is the science of constructing a cryptosystem capable of providing protection for data. Cryptography concerns with the actual protection of electronic data. This relates to the creation of systems relying on computational algorithms which offer basic data protection capabilities. The primary objective of using cryptography is to provide information security services includes confidentiality, data integrity, authentication and non-repudiation. Cryptography use different primitives includes encryption, hash functions, message authentication code(MAC) and digital signatures to provide these security services.

Conventionally, encryption techniques are used to secure data during transmission and storage and to maintain confidentiality. Encryption is also used to avoid spoofing and eavesdropping for e-commerce, wireless network security and remote access. It is possible to encrypt data, files, emails, even entire hard disks. Encryption is a reliable and effective way to protect data over the network. Modern web browsers are using the Secure Sockets Layer (SSL) protocol for secure digital transactions. SSL works with a asymmetric encryption algorithms. It uses private key for encryption and public key for decryption procedure.

The massive growth in encrypted traffic leads to a change the security landscape. As more organizations become aware of a need to secure the data, more applications, ser-

vices and devices use encryption as a prime method to protect information. Just as organizations use encryption increasingly to maintain their data confidentiality over the network, so attackers use the technology to mask their activities. This indicates encryption, which is critical to protecting confidential information in transit, such as online transactions, electronic-mails, and smart phone applications, can also permit malware concealed inside the encrypted traffic to move un-inspected via the security structure of an organization. Cyber attackers also use encryption to transmit data out of targeted organizations, largely undetected.

1.1.2 Homomorphic Encryption

Homomorphic Encryption (HE) relates to a specific type of encryption which enables computations to be performed on cipher text without needing access to a private (decryption) key.The computation outputs are encrypted and decryption can only be done by the private key holder. While conventional cryptographic algorithms can be used privately to outsource data to cloud , it is not possible to use encrypted data before even decrypting it. This leads to a massive utility failure. A cloud service, i,e. might depends on customer to download their protected records, decrypting record on local machine, and execute the required complex tasks instead of purely sending back the encrypted result to the client. The HE addresses this issue because it enables the cloud platform to preform computing while helping to protect user data with powerful cryptographic privacy assurance. The cloud only considers encrypted data, and the computation results can only be revealed by the clients.

1.1.3 Re-encryption

Re-encryption mechanisms are using cryptographic algorithm to enable third party (proxy) to modify a encrypted data for one party such that it could be decrypted by some other. A re-encryption is usually used when one party, say Sender A, intends to reveal the content of plain text sent to him and encrypted to a third party C with its public key, without proving its private key to third party C. Sender A does not want the content of his messages to be read by third party C. Sender A may designate a third party C to re-encrypt one of its messages to be sent to B. This creates a new key that can be used by B to decrypt the message. Now if Sender A sends a message to B that

has been encrypted under the key of Sender A, third party C will amend the plain text to allow B to decrypt it. This technique facilitates a number of applications such as e-mail forwarding, monitoring of law enforcement and distribution of content.

1.2 Background

Malicious use of encryption and malicious use of mathematics are evolving fields [1][2] which originated in Young and Yung's earlier research about the use of public key cryptography for designing a devoted offensive system for money extortion (Cryptovirology) [3]. However this earlier model has lot of limitations and only provides little understanding into how malware can effectively pervert mathematics and cryptography.

Encrypting malware payload prevent malware analyst, to reverse engineering of malicious code [4] and identifying malware developer's intentions. Markus Jakobosson's Asymmetric re-encryption [5] which proves the input and output encryption co-related to the exact plain text, Without leakage of information related to the plain text to verifier or the server subset of the verifier . In 2004, Golle et al [6] described a new primitive, universal re-encryption based on the Elgamal public key cryptographic algorithm, allows the re-randomization of ciphertext without knowing of the relevant private key.

A high-tech professional grade virus called Gauss [7] was detected in 2012-13 with an encrypted payload using data from the targeted victim's computer as the decryption key. No analyst can decrypt the payload and determine, what the payload will do until the virus is installed on the system of a targeted victim. Filiol presented about the encryption of malicious software [8] and described that it is feasible to stop someone to analyse the software and reversing it, likely with the use Riordan and Bruce [9] keys to encrypt payload.

In 2017, H.galteland and G.Gjosteen worked on malware encryption and randomization [10]. Malware author encrypt payload using unique key(s) generated from target environmental data, and randomize cipher text at each new node leads to form indistinguishable variant of malware in the network that infects subsequent machines or devices without the knowledge of private key. Different replicated variant of identical malware in the network boost the malware analyst's work load substantially and prevent analyst to defending some nodes in the network.

1.3 Problem Statement

In today's evolving landscape where the perimeter is almost non-existent, adopting a proactive strategy might be essential to regaining control and preventing threats in their tracks of the attack route. The best way to be confident the threat mitigation strategies will be useful to protect a firm or organizations from cyber attacks is by simulating or proactively experimenting security measures before a actual cyber attack. The traditional approach of "constructing larger fences" will no longer suffice with the growing amount and scale of cyber attacks and with the use of modern techniques by the threat actors to camouflage their malicious activities. The only way organizations can protect themselves in today's unpredictable landscape, filled with quickly evolving threat actors and innovative technologies, is by unleashing offensive cyber operations to expose aggressive adversaries on their networks.

Use of cryptographic primitives for data confidentiality and privacy is essential and important, but in the recent few years the malware authors and adversaries are continuously using the encrypted channel to perform malicious activities across the network. In particular, cyber attackers are using encryption to hide malicious activities, making it increasingly difficult to find as more organizations turn to encryption to protect data. It is important to understand the encryption schemes used by malware authors so that we can understand the defence mechanism against it, as well as use of it.

These concerns are the driving force behind our research which will enable us to design a comprehensive framework for malware encryption, re-randomization and malware propagation.

1.4 Motivation

Cyberspace is rapidly gaining importance as a war domain in addition to territory, ocean, air and space in the modern strategic discourse. States have begun to incorporate strategies and tactics in this domain to achieve reliable levels of security. Several events have also brought significant exposure to cyber-related issues in recent years. Because of the their posed threat to national security, some major cyber attacks have become a serious concern for state governments.

In addition, cyber warfare-attacks are becoming crucial since they are military and intelligence weapon which can be incorporated to existing tools in state arsenals. While no reported cyber crime that resulted mortality or physical abuse to individuals for the time being, an ever-growing number of countries around the world are preparing for conflict. In this context, national doctrines, cyber-defence tactics and defensive and offensive cyber-warfare capabilities have been developed.

In the era of 5th generation war (cyber warfare), the terrorists and law enforcement agencies uses cyber technology, both as an offence and defence means. Today, encryption provides security and confidentiality and protects individual and organizational communication from unsophisticated and sophisticated offenders and repressive government. So, encryption is used as a major defence mechanism. Our law enforcement agencies can use this encrypted traffic used by opponents to hide the encrypted malware and can attack the target in secure environment.

While the number of domestic crime are increasing, malware can be helpful to unmask the identification of criminals who essentially programmed their machines to prevent from identification. Malware encryption is much more effective and often used to identify or track opponents that have shielded their operations or connect via reliable encryption procedures.

1.5 Objectives

The goal of our research work is to establish a proof of concept for the malware encryption schemes using asymmetric cryptography. Our aim is to evaluate the propagation of malware and how malware propagation can be prevented from analysis. Our research have following key objectives:

1. To evaluate malware encryption schemes using asymmetric homomorphism for randomization cipher text to protect certain malware payload and with provision of efficient use of environmental keys for encryption and decryption process of these payloads.

2. To implement a framework for malicious software encryption, re-encryption or re-randomization to create encrypted payloads such that any two malware samples are indistinguishable from each other.

3. To propose a novel scheme for encryption and re-randomization of malware payload based on asymmetric algorithm, other than ElGamal.

4. To evaluate different methods and defensive approaches that can be used for detection and identification of malicious communication in the encrypted traffic.

1.6 Thesis Organization

In this chapter, introduction, problem statement, background of research work, motivation and objectives of this research have been clearly explained.

The rest of the thesis is structured to provide a firm background on malware encryption, propagation, and re-randomization. This includes existing research on malicious payload encryption and re-encryption as well as presenting the novel concepts that this thesis has proposed.

Chapter 2, presents the background information and preliminaries involved in the thesis. We first present the malware and its different types. After that, we introduce the cryptography, symmetric and asymmetric cryptography. At last, we describe the homomorphic encryption and well-known partially homomorphic encryption scheme algorithms include RSA, ElGamal and Paillier cryptosystem.

Chapter 3, includes the literature review by which the academic research conducted by different writers in the context of malware and offensive use of cryptography is discussed and justified. First, we introduce the offensive use of cryptography for malicious purposes. After that, we present a detail review on proxy re-encryption schemes and the research work already carried out in the domain. We describe the use of environmental keys for encryption and decryption purposes.

Chapter 4, explain the malware encryption algorithm with the use of malware samples. Initially, we introduce malware encryption with their features and significance. Secondly, we describe the malware propagation framework for malware author or creator as well as for malware analysts. In addition, we illustrate the malware encryption process, the use of asymmetric ElGamal cryptosystem and environmental keys for malware encryption. Furthermore, an existing scheme which use homomorphic encryption of ElGamal to re-encrypt the malicious payload to generate the indistinguishable malware variants of same malware, will be described. At last, we propose our implementation of existing ElGamal

based mawlare encryption and re-randomization scheme and experimental results.

Chapter 5, presents our proposed scheme for malware encryption and re-randomization based on asymmetric cryptosystem, Paillier. Initially, we introduce an overall malware potential, homomorphic property, encryption and decryption process of Paillier's algorithm. We also discuss the use of the PHE scheme for re-randomizing malware and probabilistic and deterministic encryption. We illustrate the principle of semantic security and drawback of RSA cryptosystem for malware re-randomization. At last, we presented our proposed scheme for malware encryption and re-randomization to propagate malware, built on the Paillier asymmetric cryptosystem and the experimental results.

Chapter 6, provides an overview of the approach used by malware analysts to detect and analyse encrypted malware. We first identify the latest malware trends and the global scale of malware. Thereafter, the widely used malware detection techniques for encrypted malware, including entropy-based detection, signature-based malware identification, hidden Markov-based analysis, machine learning based detection and Cisco ETA solution are presented.

Chapter 7 aim is to provide a review of previous chapters in order to conclude the research work. This provides an overview of malware encryption and re-randomization scheme based on the public key cryptographic algorithm ElGamal and also highlights our contribution to this work. We also summed up our proposed work on Paillier based malware encryption and re-randomization. At last, we mention the future direction of research.

Chapter 2

Background

2.1 Introducation

In this chapter, we explain the background and preliminaries involved in the research work. We first present the malware and its different types. After that, we introduce the cryptography, symmetric and asymmetric cryptography.

At last, we describe the homomorphic encryption and well-known partially homomorphic encryption schemes include RSA, ElGamal and Paillier cryptosystem.

2.2 Malware and its Classification

The word malware, shortened for malicious software, define as a computer program which executes for the adversary on any target system without the system owner's authorization. Malware is highly popular among security researchers and cyber criminals as it provides attractive potential for revenue. This popularity makes malware a significant threat for the digital industry. Because malware research is an interdisciplinary and complex field of study. In this section, we describe some significant terms and ideas for clarity purposes.

There are different methods to classify malware into certain categories based on specific features, including transmission, inflammation, obfuscation, ex-filtration, command and control or Camouflage strategies, the activities performed on the operating system (OS) during run time. In addition, it is becoming extremely hard to define categories of malware as nowadays malware developers can readily reach multiple malware samples

source code and combine their functionality to generate latest and robust versions. In addition, having an update procedure to extend their capacities is becoming increasingly common with malware samples. For instance, one sample can exfiltrate the credit card information and credentials of the customer while attaching a plugin to the OS to obtain legitimacy at system level. Although there is no overall agreement on the taxonomy of malware, popular types of malware and their applications can be summarized as follows:

2.2.1 Virus

A virus is a type of malware which uses some other software to be activated. It is able to replicate itself. However, normally pursues any network-related tasks, like infecting some other victim, exfiltrating data from compromised computer, etc. Many viruses have been created to exploit the operating system or perform quite destructive operations on the operating system Whereas some are innocuous and produced for credibility in politics. Since computer viruses are really the earliest known malware , vast majority of individuals nowadays use the word virus to describe every kind of malware. Due to the widespread human use term virus, prominent data protection and cyber security firms tend to use the virus term to name their products, for example Virustotal.

2.2.2 Worm

Worm is type of malicious software which operates independently; there is no need for a host program to deployed(despite intervention). A worm seems to have the ability to replicate through the penetration of built-in OS services and loopholes and also external networking services via computer networks. The other technique used by worm to propagate itself is through social engineering. Social engineering tricks clients through carrying out voluntary conduct like filling and returning bogus websites, login details, attempting to open harmful appended document or email etc. Worm also used network applications (internet server, file sharing) to leverage misconfiguration. Ultimately, worms used the default username and password pair brute force attack.

Because in behind the scene the spread of worms occurs silently, the target is generally unaware of the attack. In several circumstances, immediately following penetration, worms execute malicious payloads. Conficker is among the most common worms in malware history that target MS Windows operating system, often referred to as Kido.

Conficker's first-ever dataset was identified in Oct 2018 and hundreds of thousands of computers were infected globally. Conficker exploits the vulnerability of Windows OS ' built-in network features, with every release from MS Win-2000 to Windows Server-2008 operating systems for Internet propagation.

2.2.3 Trojan

Trojan is a malicious software that appears to be extremely beneficial to clients and also tends to motivate installation. Such application, however, often contains concealed binary executable that can impact after deployment and therefore could result to various unwanted effects. Because the application functions properly, finding out the program's consequences is very hard for a normal computer user. Today's trojans have very advanced capabilities to capture all key strokes from gaining control of OS including all processes. For example, Poison Ivy [11], is a popular trojan which provides the opponent with complete control over the computer of the compromised client. Poison Ivy had first been discovered in 2005 and it is also renowned as the Remote Access Trojan (RAT). Once trojans are activated, key logging, screen shooting, camera video recording, and critical information sniping networks can be performed.

2.2.4 Backdoor

Backdoor is the malicious tool that an attacker on the target system has installed to gain remote access. Cyber criminals take advantage of different vulnerabilities to install backdoor on victim machines. They sometimes trick customers into installing backdoor by making them think the program is valid. They use distinct methods after installation to frequently restart backdoor. Some of these include editing startup files, removing a registry key which is used every time the system starts up and does the job as a planned task. Certain backdoor allow attackers to alter their root or administrator privileges enabling command remote execution, and allow attackers to monitor any target system activity.

2.2.5 Rootkit

Rootkit is a program that combines the Trojan horse and backdoor behaviours together and also alters other operating system programs [12]. They exhibit Trojan behaviour, which allows attackers to access a system remotely by replacing the original version of a file with a contaminated copy and backdoor behaviour. It also changes operating system programs in contrast with Trojans and backdoor. Rootkits are divided in two kinds according to the operating environment, one is User Mode Rootkits and the other is Kernel Mode Rootkits. User Mode Rootkits substitute apps with malicious code on top of the kernel to accomplish their objective. This hides an adversary existence. Rootkits in kernel mode are same like rootkits in user mode with the exception of changes in operating environment. In this situation, they change the victim's kernel itself completely.

2.2.6 Spyware

Spyware is a program that gathers secret user information, records web navigation activities and gives all data for financial advantages to a third party [13]. ActiveX checks, plug-ins and program exe are the most commonly used methods for spreading spyware. In reality, the simplest and most efficient way for adversaries to propagate spyware is to use ActiveX controls. A plug-in is a program that improves the features of the web browser. The plug-ins are downloaded and installed on the web browser by ActiveX.

2.2.7 Adware

Adware is recognized as programs showing ads in pop-up shape, flash and other media in the customer interface. Some adware forms also behave as spyware and gather private information. Programmers are often paid for by commercial organizations to write these types of programs.

2.2.8 Ransomware

Ransomware is the malicious program that is executed to restrict the customers against the cessation of computer resources and calls for ransom to release restrictions. Some

ransomware encrypts important system files, whereas several affects their credentials or locked system. Crypto-Locker [14] is among the most well-known ransomware that is targeted at Windows Operating system. By blackmailing customers through social engineering to execute a e-mailed document, CryptoLocker propagates. When the mailed document is opened by user, CryptoLocker activated itself, and by RSA encryption algorithm, it encrypt the sensitive documents or all hard disks of the system.

2.3 Cryptography Preliminaries

Initially the term cryptology comes from kryptos, logos and which mean "concealed word". Cryptology is generally a science that explores how customer data can be protected. Cryptology is split into two distinct areas, cryptography and crypt analysis, where cryptography is the study of the design of secure ciphers and has a defensive role in information theory whereas the crypt-analysis is science of breaking cipher and testing strength of cryptographic algorithms. The purpose of cryptography is to prevent unauthorized individuals from accessing any private data by offering some of the following characteristics:

- Confidentiality relates to protecting data from unauthorized individuals being assessed. In other words, only those legally permitted to do so can access sensitive information. Just think of our bank accounts
- Integrity is the protection of data against unauthorized entities being altered.
- In non-repudiation the creator or sender can not refuse its intention in the creation or communication of the information in a later stage.
- In authentication the sender and recipient can verify the identity of each other and their origin and destination.
- Availability of data means ensuring that approved persons can access data if required.

Modern cryptography is relying strongly on theory of mathematics and computer science. Cryptographic algorithms are formulated around assumptions of computational hardness which makes it difficult for any opponent to defeat such algorithms practically.

In cryptography, encryption is the mechanism of encoding a text or data so that it can only be accessed by authorized parties to ensure privacy. Imagine Alice intends to send Bob a confidential message, but she is worried that Eve will intercept the message and read the message. Alice encrypts the message in a manner that only Bob can decrypt to the private message.

Cryptography falls into two major branches, symmetric cryptography and asymmetric cryptography. Symmetric ciphers are generally known secure when there is no crypt analysis attack that can defeat them more effectively than exhaustive search. The complete absence of such an attack, however, is not proven to be in general. As a consequence, level of trust in a symmetric cipher's security generally results in several years of public existence without crypt-analysis. On the other side, asymmetric cryptosystems are generally based on computationally complex mathematical problems from the concept of number theory.

2.3.1 Symmetric Cryptography

Symmetric key cryptosystems use the exact key for encryption as well as decryption. This is the fundamental concept, out of which derives the term "symmetric." The sym-

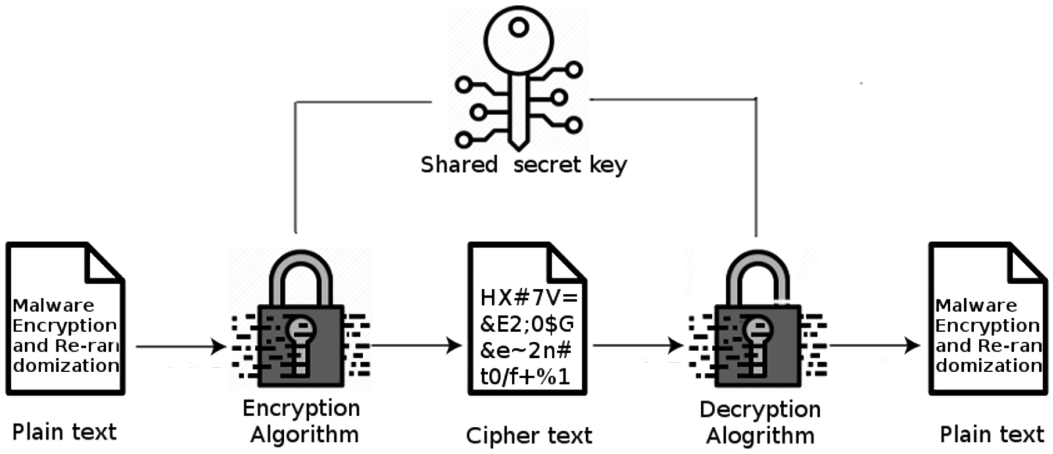

Figure 2.1: Symmetric key cryptography

metric key ciphers are further divided into two subgroups, the block cipher and stream ciphers. A block cipher is type of cryptographic encryption that simultaneously encrypts a fixed size of data bits, called block. Each block's usual size is 64, 128, 256 bits. A 64-bit block cipher, for example, will take 64 bits of plaintext and encrypt it into 64

bits of cipher text. Padding schemes are called into play in cases where plaintext bits are shorter than block size. Most of today's symmetric ciphers are block ciphers. Most of the widely used encryption schemes that fall under the category of block ciphers are Data Encryption Standard(DES), Triple DES, Advanced Encryption Standard, IDEA, and Blowfish etc.

A stream cipher encrypts messages using a pseudo random key stream. Every bit of the message with the corresponding key stream digit is encrypted bit by bit. In cases where speed and simplicity(real time application) are both requirements, stream ciphers are typically used. When a 128 bit block cipher like AES were used instead of a stream cipher where 32 bit block messages were encrypted, there would remain 96 bits of padding. This is an ineffective strategy and one justification why a stream cipher is preferred because it performs on the smallest unit possible. Some popular stream ciphers include, RC4 , Salsa20, ChaCha, Rabbit, HC-256 etc. In stream mode, block ciphers can be used to work as a stream cipher. If a block cipher is running in CFB, OFB or CTR mode, no additional measures are required to handle messages that are not equivalent to multiples of the block size and avoid the padding effect.

2.3.2 Asymmetric Cryptography

Asymmetric cryptography was introduced to counter the drawbacks of symmetric cryptography. This method is also known as public-key cryptography as it does not require prior key transfer between sender and recipient. Key pair (private key and public key) is used to archive the objective. Key generation algorithm are use to generate key pair, Both public and private key relates each other using mathematical methods. How to generate the keys, its depends on the algorithm. Keys can be used in two aspects: delivering a secure message and proving authorship. Message is encrypted with public key receivers in the first case and can only be decrypted by these private key marchers. For data confidentiality, this method is used. To prove authorship, a message should be encrypted with the private key. In this scenario, it would be possible for everyone to decrypt and make sure that message belongs to a person whose public key decrypt it.

Rivest, Shamir and Adleman (RSA), Digital Signature Algorithm (DSA), Diffie-Hellman, Rabin, ElGamal etc. are the most well-known public key ciphers.

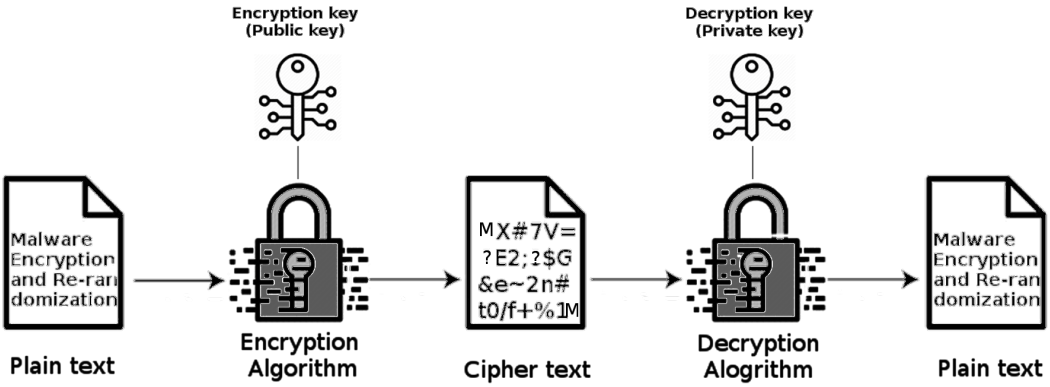

Figure 2.2: Asymmetric key cryptography

2.4 Homomorphic Encryption

Homomorphism is defined as a feature in abstract algebra which maps an entry item from the domain set to an item in the range of an algebraic set. Homomorphic Encryption (HE) is a cryptographic encryption mechanism that allows a cloud service platform to execute specific data computing operations when encrypted. Clients have to compromise their data confidentiality with traditional encryption schemes to use the cloud services, as conventional schemes can not permit the provider to function on encrypted data. Consequently, a scheme needs to be developed in which computation functions can still be carried out on encrypted data.

In 1978, Rivest, Adleman, and Dertourzous first proposed the idea of the use of homomorphic encryption to make some computations on encrypted data after the RSA scheme [15] . It was named "Privacy Homomorphism" [16]. Almost all the schemes proposed have been able to operate on encrypted data with one or few operations. The advanced milestone occurred with Gentry's ideallatisti [17] system, which can carry out an infinite number of arbitrary function operations. But due to its high computation costs, it really was not a practical scheme. New optimizations and advancements were proposed in subsequent years with the other schemes.

Depending on the computational operations, there are three various categories of HE schemes, which are:

- Partially Homomorphic Encryption (PHE) partially supports one type operations

performed as many times as required.

- Somewhat Homogeneous encryption (SWHE) supports only a few kinds of functions for only several times.
- Fully Homomorphic Encryption (FHE) supports every type of operation carried out as often as desired.

We begin with the fundamentals of homomorphic encryption. Then describe the famous schemes with examples.

2.4.1 Homomorphic Encryption Schemes

Homomorphic encryption is a specific type of encryption, is capable of performing computation on cipher text, and results out of the same computation performed on the plain text, as shown in figure 2.3. The Homomorphic encryption , with encryption algorithm *Enc*, over an operation '*' can be defined as:

$$Enc(m) * Enc(m') = Enc(m_1 * m'), \quad \forall m, m' \in M$$

Where M is termed as message space [18].

There are four main algorithms for the homomorphic encryption(HE) scheme:

- **The Key Generation Algorithm,** *KeyGen*: take a security parameter as input. For symmetric HE scheme output a secret key, and for a asymmetric HE outputs pair of private key and public key.
- **The Encryption Algorithm** *Enc*: takes input message m from message space M and encryption key k from *KeyGen*. It outputs cipher text c from an underlying cipher text space C, as:

$$c = Enc(m)$$

- **Decryption Algorithm** *Dec*: takes a decryption key and cipher text c and generates an output message m, as:

$$m = Dec(c)$$

- **The Evaluation Algorithm** *Eval*: takes input cipher text (c, c'), perform some operations $f()$ on cipher text to get evaluated output cipher text, without knowledge of both plain text and secret or private key. The evaluated cipher text is:

$$f(c, c') = E(f(m, m'))$$

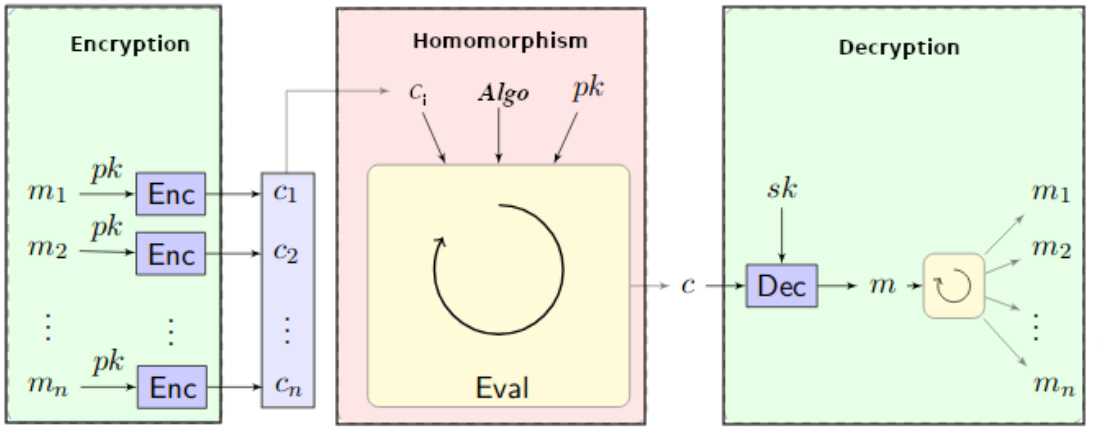

Figure 2.3: Homomorphic Encryption Scheme

It is important to preserve the cipher text format after an evaluation process, to accurately decrypt it. The size of the cipher text should also remain constant in order to execute infinite amount of operations. The size of the cipher text would otherwise be increased, limiting the number of operations conducted. In PHE systems, the Eval function supports only addition or multiplication, while supporting limited amount of SWHE operations. In FHE systems, Eval supports the evaluation for infinite amount of times of arbitrary functions over cipher text.

2.4.2 Partially Homomorphic Encryption

Several conventional cryptographic encryption schemes can be designated as PHE, because they can conduct a single type of computation on encrypted data.

Partially homomorphic encryption (PHE) is either a homomorphic additive system that only allows additive processes or multiplicative homomorphic scheme that only allows multiplicative operations on encrypted information.

We discussed some important and well-known partially homomorphic encryption schemes, such as, RSA [15] , Elgamal [18] and Paillier cryptosystem [19].

2.4.2.1 RSA

Rivest, Shamir, and Adleman [15] published RSA in 1978,l soon after Diffie Hellman [20] introduced public key cryptography in 1976. The security of RSA scheme was relying on the integer factoring problem of two large prime numbers. Rivest, Adleman and Dertouzous subsequently published the homomorphic property of RSA with the name "privacy homomorphism" [16], an early example of PHE.

	ADD-HE	MUL-HE	MIXED
RSA	×	✓	×
ElGamal	×	✓	×
Paillier	✓	×	×

Table 2.1: Comparison of various schemes for Homomorphic encryption .

There are four main algorithms for RSA partially homomorphic scheme(PHE) scheme:

- **Key Generation Algorithm**(*KeyGen*): The key generation process outputs a public key(pk) and a private or secret key(sk). The public key comprises two numbers e and n, where n is a product of two big primes p and q, chosen by the owner of the secret key. The second integer e has to be selected in such a way that the greatest common divisor of e and $\phi(n)$ is $gcd(e, \phi(n)) = 1$. i.e. e is invertible mod $\phi(n)$. Here $\phi(n)$ is the Euler Totient function [21]. The secret key(sk) is (d, n), where d is computed to be the inverse of e, i.e. $ed \equiv 1 \mod \phi(n)$. This can be achieved by using extended euclidean algorithm [22].

- **Encryption Algorithm** *Enc*: The encryption algorithm takes a message m as a input from the message space Z_n and computes the cipher text c as:

$$c = m^e \mod n$$

  Without knowledge of p and q, the integer c, can not be traced back to the initial message ($c \in$ cipher text space Z_n).

- **Decryption Algorithm** *Dec*: Decryption algorithm takes cipher text c and secret key (d, n) as input and compute the message m as:

$$m = c^d \mod n$$

  Because d is the inverse of e, this is indeed the initial message.

- **Homomorphic Property** *HE*: RSA has a homomorphic multiplicative property. This implies multiplications can be performed with encrypted messages without losing or manipulating their original data.

  For input message $m, m' \in Z_n$,

$$\begin{aligned} Enc(m) * Enc(m') &= (m^e \mod n) * (m'^e \mod n), \\ &= ((m * m')^e \mod n), \\ &= Enc(m * m'). \end{aligned}$$

  From the above equation we can see, RSA's multiplicative homomorphic property can evaluate $Enc(m * m')$ directly from $Enc(m)$ and $E(m')$ without decrypting them.

2.4.2.2 ElGamal

In 1985, Taher ElGamal [18] introduced a new public key cryptosystem based on the difficulty problem of the discrete logarithm [23]. The ElGamal is regarded an advanced Diffie-Hellman algorithm variant [20]. ElGamal is used mainly for encrypting the secret key of symmetric cryptographic system. The ElGamal public key cryptosystem algorithm are:

- **Key Generation Algorithm** *KeyGen*: Choose a finite group G of order n, with a generator g. Compute $y = g^r$, for some uniformly chosen random integer $k \in Z_n$. The KeyGen algorithm outputs the public key (G, n, g, y) and private key k.

- **Encryption Algorithm** *Enc*: The encryption algorithm takes a message $m \in G$ as an input, pick a random integer $r \in Z_n$ and computes the pair of cipher text c_1 and c_2 as:

$$\begin{aligned} c_1 &= g^r \mod n \\ c_2 &= m \,.\, y^r \mod n \end{aligned}$$

- **Decryption Algorithm** *Dec*: Decryption algorithm takes cipher text pair c_1, c_2 and private key k as input and compute the message m as:

$$m = c_2 \,.\, c_1^{-k} \mod n$$

- **Homomorphic Property** *HE*: The ElGamal public key cryptosystem has also a homomorphic multiplicative property. This implies multiplications can be performed with encrypted messages without losing or manipulating the original data.

  For input message $m, m' \in G$,

$$\begin{aligned} Enc(m) * Enc(m') &= (g^r, m.y^r) \ * \ (g^s, m'.y^s), \\ &= (g^r.g^s, \ m.y^r. \ m'.y^s), \\ &= (g^{r+s} \ , \ (m.m')y^{r+s}), \\ &= Enc(m * m'). \end{aligned}$$

2.4.2.3 Paillier

The Paillier encryption scheme was introduced by Pascal Paillier in 1999 [19]. The algorithm of the Paillier's system is constructed on composite residuosity problem [23], which is known to be computationally hard to solve. It is a probabilistic asymmetric algorithm that inherits the additives homomorphic properties for public-key cryptography. The Paillier cryptosystem's algorithms are:

- **Key Generation Algorithm** *KeyGen*: The Paillier key generation algorithm takes two big prime integers p and q as input such that:

$$gcd(p.q, \ (p-1)(q-1)) = 1$$

  Next is to compute composite $n = pq$ and λ as:

$$\lambda = lcm(p-1, \ q-1)$$

  Choose an integer $g \in Z^*_{n^2}$ such that:

$$gcd(n, \ L(g^\lambda \mod n^2)) = 1$$

$$Here, L(x) = (x-1)/n, \ \forall x \in Z^*_{n^2}$$

  The outputs of Paillier KeyGen algorithm are public key pair (n, g) and private key pair (p, q).

- **Encryption Algorithm** *Enc*: The Paillier encryption algorithm takes a message $m \in Z_n$ as input and randomly selects an integer $r \in Z^*_n$, this random number is

being used to satisfy the probabilistic property of paillier that one plain text can have several cipher texts. This random variable does not interfere to decrypt the cipher text correctly, but changes the corresponding cipher text.

The output of encryption algorithm is cipher text $c \in Z_{n^2}$ that has the following form:

$$c = g^m.r^n (\text{mod } n^2)$$

- **Decryption Algorithm** *Dec*: Decryption requires cipher text c and private key p and q as input. The message is computed as:

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)}$$

Due to the condition that n and $L(g^\lambda \mod n^2)$ are co-prime, it is possible to find inverse of $L(g^\lambda \mod n^2) \mod n$.

- **Homomorphic Property** *HE*: The Paillier public key cryptosystem has a homomorphic additive property. This implies addition operations can be performed with encrypted messages without losing or manipulating the original data.

For input message $m, m' \in Z_n$,

$$\begin{aligned} Enc(m) * Enc(m') &= (g^m.r_1^n \mod n^2) * (g^{m'}.r_2^n \mod n^2), \\ &= (g^{m+m'}.(r_1 * r_2)^n \mod n^2), \\ &= Enc(m + m'). \end{aligned}$$

This means that the encryption of two plaintext m and m' is precisely the multiplication of the relevant cipher texts c and c' AS mention above that Paillier is homomorphic over addition. But it has also some additional operation over plain text m, m' as:

$$\begin{aligned} Enc(m) * Enc(m') \ (\text{mod} n^2) &= Enc(m + m') \ (\text{mod} n) \\ Enc(m) * g^{m'} \ (\text{mod} n^2) &= Enc(m + m') \ (\text{mod} n) \\ Enc(m)^k \ (\text{mod} n^2) &= Enc(km) \ (\text{mod} n) \end{aligned}$$

Chapter 3

Literature Review

3.1 Introduction

In this chapter, we present a general review of the related research. First we introduce the offensive use of cryptography for malicious purposes. After that, we present a detail review on proxy re-encryption schemes and the research work already carried out in the domain. We describe the use of environmental keys for encryption and decryption purposes.

3.2 Offensive use of Cryptography

Cryptography, on one side, is the most crucial aspect and significant mechanism for protecting cyber space, but, at the opposite side, cyber terrorists can also unlawfully use it to target digital communications or engage in some criminal activity in cyberspace, in this case, cryptography (especially when using worms, Trojan horses and back doors) became a hazardous cyberspace threat.

Offensive use of cryptography starts with Young and Yung work [24] however, with a quite small vision. By using this technique, a malware that is used to data encryption of client using asymmetric cryptographic algorithm's public key and extorts money from the clients who desperately wants the key(private key) to get back to their data. The malware on its own does not use malicious mathematics or malicious cryptology in this strategy. Just the payload utilizes cryptology to extortion money. Young and Yung's input does not propose more than just a cryptographic book, although it has the value

to initiate it. Their contribution results in a fascinating yet disturbing new fields of academic research and experiments. This became a building block for use of cryptographic encryption for malicious use. In 2005, Filiol [25]) describes the first scholarly description of the using cryptography in the important process itself. In this case the public key cryptography, together with the environment key management, allows malware code to be protected against reverse engineering for particular operational conditions.

After that lot of academic research work was published to protect malware using cryptographic techniques. In 2007, Filiol et al published a paper in which he described optimized malware(worm) propagation and different attack methods [26], for example , using particular random number generator

Another work of Filiol [8] describe optimal methods of self-protection and how malware code protects itself and its functional operations with very secure and robust symmetric cryptographic algorithms. Filiol and Josse [27] proposed the Partial or complete invisibility characteristics. By using statistical simulability of Filiol and Josse, the software developer aims to create his script or code invisible. Eric Filiol have many great contribution in the malicious use of mathematics. In 2008, he published another paper in which he discussed the use of the concept of difficulty or computeability number theory to create undetectable malicious software [28]. To protect binary executable files from reverse engineering, Filliol proposed the methodology [4] that uses cryptographic techniques or encryption algorithms with different approaches.

3.3 Re-encryption Schemes

Re-encryption mechanisms are cryptosystems that enable third parties (proxy) to modify a encrypted cipher text for one party, so that it could be decrypted by some else, as in figure 3.1. A re-encryption is usually used when one party, say Sender A, intends to reveal the content of plain text sent to him and encrypted to a third party C with its public key, without proving its private key to third party C. Sender A does not want the content of his messages to be read by third party C. Sender A may designate a third party C to re-encrypt one of its messages to be sent to B. This creates a new key that can be used by B to decrypt the message. Now if Sender A sends a message to B that has been encrypted under the key of Sender A, third party C will amend the plain text

to allow B to decrypt it. This technique facilitates a number of applications such as e-mail forwarding, monitoring of law enforcement and distribution of content.

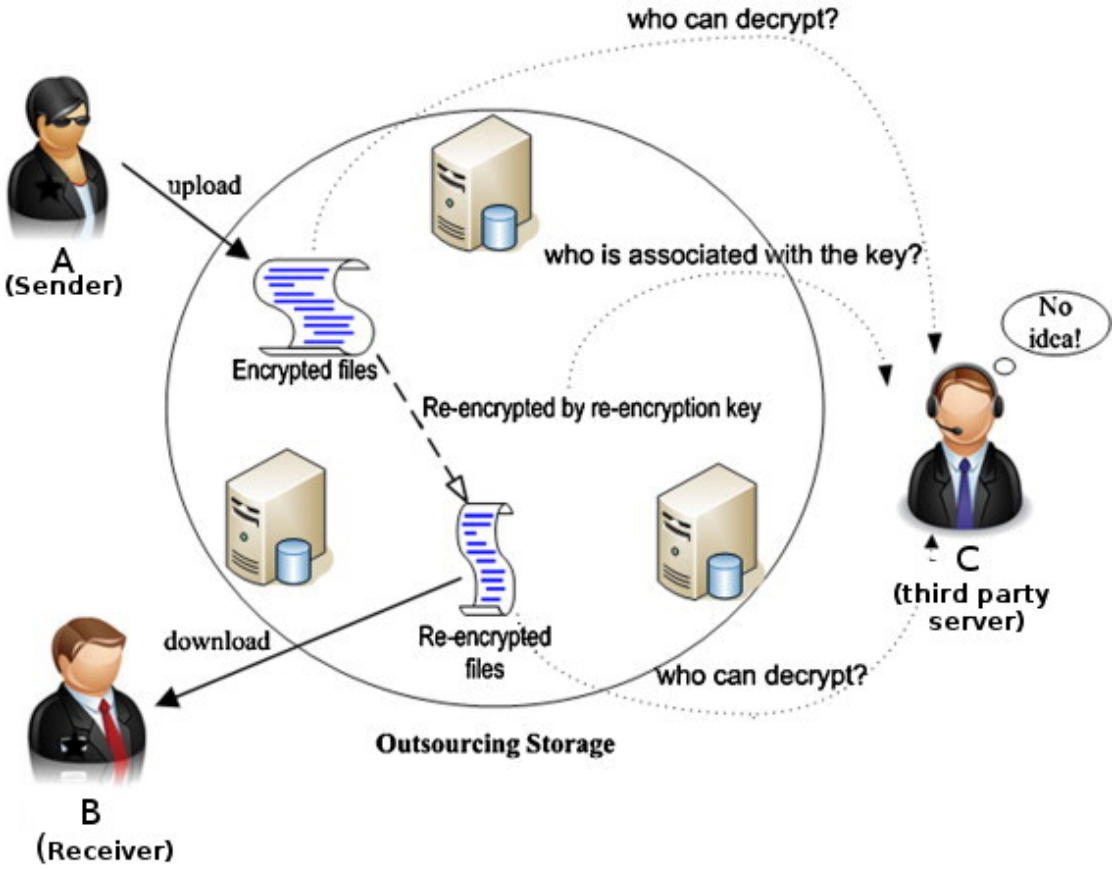

Figure 3.1: Proxy Re-encryption

Mambo and Okamoto [29] initially proposed a technique for delegate decryption privileges primarily as an increase in productivity across conventional decryption and re-encrypt techniques. The concept of "atomic proxy cryptography", in which a third party uses a procedure that transforms Alice cipher text into a new version of cipher text for Bob without knowledge of plain text or secret key, was suggested by Blaze, Bleumer and Strauss in 1998 [30]. Their another work that is based on ElGamal public key cryptosystem, the proxy is given a delegate key $\frac{b}{a} \mod q$ in order to convert cipher texts from Alice to Bob by means of computing $(mg^k \mod p,\ (g^{ak})^{\frac{b}{a}})$ using a secure prime $p = 2q + 1$ modulus. However, the publishers notice that somehow this model has an intrinsic limitation, It's in two ways , as to say, the result of equation b/a is being utilized for transfer the cipher texts from Alice to Bob. Therefore, this model is only beneficial if Alice and Bob have shared trust relationships. The BBS scheme's delegation is merely

descriptive, meaning that the proxy itself can set up delegate privileges for both two entities that did not agree about this. For instance, a/b and b/c values allow the proxy to encrypt again(encrypt cipher text) data from Alice to Carol. A further shortcoming of this method is the fact, once the proxy and Bob join together, then they also can retrieve their private key, like $(a/b) * b = a!$.

Jakobsson [5] has formed a protocol based on quorum, dividing the proxy into sub-parts, each of which controls the re-encryption key. Here, the delegator keys will be secure as soon as some proxies are reliable. Zhou, Mars, Schneider and Redz took a comparable strategy [31] into consideration.

Recently, Dodis and Ivan [32] have published unidirectional proxy encryption for Elgamal, the RSA and IBE cryptosystem, through the exchange of a private key between the two entities. The private keys of Alice are split into two groups s_1 and s_2, Where $s = s_1 + s_2$ in their unidirectional ElGamal system and circulated to the proxy and to Bob. The first proxy computes $(\frac{mg^sk}{g^k})$, which Bob can decrypt as $(\frac{mg^{s_2k}}{(g^k)^{s_2}}) = m$, if received by cipher texts of the forms (mg^{sk}, g^k). Although this methodology has several benefits over BBS, new pitfalls are also highlighted. These "secret-sharing" methods don't alter Alice's ciphertext into Bob's cipher texts, which is in the broadest sense, such that Bob will decipher them with his own private key). In reality, they delegate decryption, which can be hard to handle for him, by requiring Bob to hold additional keys. One area of concern is the Dodis-Ivan IBE [32] proposed system in which the master key is transmitted between the proxy and the delegate that decrypts all cipher texts. The delegate therefore needs to hold only master key, but an obvious flip side is that the proxy as well as every delegate in the scheme can decrypt data from anyone else.

In 2004, Philippe Golle introduced a cryptographic scheme known as "Universal re-encryption of mixnets" [6]. The scheme is based on conventional ElGamal cryptosystem. The idea is to re-encrypt or re-randomize the ciphertext without the knowledge of corresponding private key. This scheme become building block for re-encryption process, based on asymmetric cryptography.

3.4 Environmental Key Generation

Environmental keys refers to the generation of data from the computer system and network triggers, for example IP address, path variable etc. But the collected environmental data may be raw data with arbitrary length and have different format. To generate encryption or decryption keys from this data, need to transform it to a standard format and, may be a fix length. The final key must be regenerated from the environmental variables.

Bruce Scneier, is the first who introduced the concept of use of environmental key for cryptographic encryption or decryption process to use it in mobile agents [9]. Keying content which is created from different categories of collected environmental content. The mobile Agents may obtain encrypted data using these keys that can only be decrypted if certain environmental variables are valid. Agents withdrawn or executable data encrypted with these keys could remain unsure from content of data, until certain environmental conditions have been encountered.

For instance, a high-tech professional grade virus called Gauss [7] was detected in 2012-13 with an encrypted payload using data from the targeted victim's computer as the decryption key. There was various payload variants. They hold different extras PE sections, which was encrypted with RC4 stream cipher. The RC4 encryption key used to encrypt sections, contain combination of two environmental variable generated from system. The first variable was environmental string $PATH$ (in windows OS), the other one was the name of directtory in $PROGRAMFILES$. No analyst can decrypt the payload and determine, what the payload will do until the virus is installed on the system of a targeted victim.

Malware requires to running on a particular target system while at the same time, need to maintain its payload anonymous from a malware analyst. Generally, conventional cryptographic encryption algorithm can be used to achieve confidentiality, but in the case of maintaining the secrecy of key , it is a different scenario. Because, malware need to execute on the target machine. Hence, the secret key will in plain text format. If the secret key is stored in the malware to decrypt the payload, the malware analyst can extract the key. The environmental keys can be used to encrypt data, so that later on it can be used in decryption process.

Chapter 4

Malware Encryption and Re-randomization

4.1 Introduction

In this chapter, we evaluate the malware encryption scheme and describe their practical use for malware samples. First we introduce the malware encryption, the features and objective of malware encryption. Secondly, we describe the malware propagation model with respect to both malware author or developer and malware analyst. Further, we describe the malware encryption process, the use of asymmetric encryption, and environmental keys in context of malware encryption. Further more, an existing which use homomorphic encryption of ElGamal to re-encrypt the malicious payload to generate the indistinguishable malware variants of same malware, will be described. At last, we explain the technical detail of our implementation of ElGamal based malware encryption and re-randomization scheme and implementation results.

4.2 Encrypting the Malicious Payload

The malware author's objective is to prevent the analysis of malicious code and to obscure the malware author's intents from analyst, that defending some computer nodes on the network. To achieve this goal malware developer use encryption techniques. The malware developer wishes to target a specific computer on the network. The author then begins with infecting X initial nodes, with X distinct variants of his malicious

program. Through making replicas of the initial version, malicious software continues to compromise successive network computers. On the other hand, the goal of the malware analyst is to identify the compromised computers in the network with some malware and to desired to know that any of the malware targets his network devices. So, the malware analyst only needs to know initial X malware samples.

To increase the malware analyst's workload, H.galteland and G.Gjosteen worked on malware encryption and randomization schemes [10]. The malware author encrypt payload using unique key(s) generated from target environmental data, and re-randomize cipher text at each new node leads to form indistinguishable variant of malware in the network that infects successive machines without the knowledge of private key. Different replicated variant of identical malware in the network grows the malware analyst's work load substantially and prevent analyst to defending some nodes in the network. This scheme is based on public key encryption scheme ElGamal that allow re-encryption of cipher text. The re-encryption exploits the homomorphism of ElGamal cryptosystem, to re-randomize the cipher text. Re-encryption is a simple enhancement of the ElGamal symmetric encryption that allows the re-randomization of cipher text despite knowing the relevant secret key.

4.3 Malware Propagation Model

The malware developer's goal is to target a specific computer or node in the network. The malicious developer's also need to resist against reverse engineering of path towards the source of the malware. The malware author also wants to hide attack motives and origin of malware from opponent.

On the other side, the malware author's opponent is a malware analyst, whose job is to protect and defend the node in the network from malware attack. His objective is to identify the malware sample that targeting his node in the network. The analyst also wants to identify the source of malware and determine the intent of malware authors. The goal is to hide the intents and identity of malware author from analyst by reverse engineering towards source path.

We use the malware propagation model, shown in figure 4.1, in which the malware author with malware M and the malware source M_{source}, infects X initial peers or nodes(in or

outside the target network) with distinct variants of his malware. In response the X initial nodes infects subsequent machines in the network by propagating in-distinguished copies of same malware M.

Each and every direct connection to the malware source helps to increase the malware analyst's probability of finding the malware author's origin, because of that the malware author must preform as several additional infections as feasible and prefer indirect routes to the target node T.

The responsibility of the malware analyst is to protect computers in the network N from any potential threat of malware, and he has comprehensive understanding of the environment that he protects. The malware analyst can observer the wider malware space M_s, to find the more malware M samples.

The malware author will use cryptographic algorithms to encrypt the malicious payload to make the work of the analyst more challenging. Encrypting the payload protects malicious software code from being reversed and tries to obscure the malware author's intents.

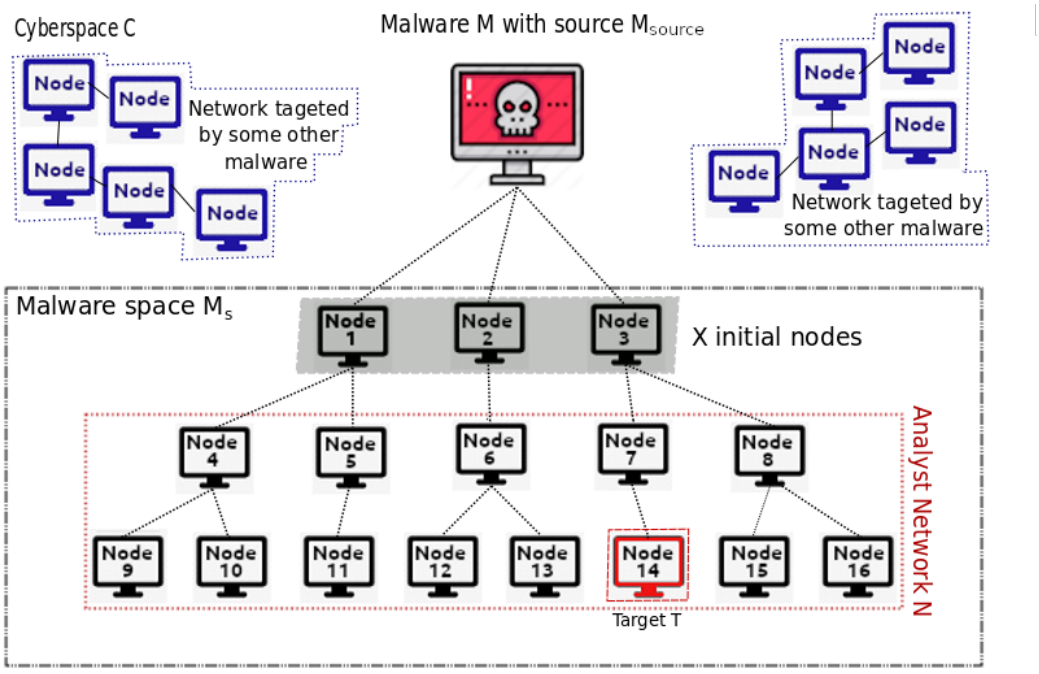

Figure 4.1: Malware propagation model

4.3.1 Malware Propagation Modeling

The malware propagation modeling is based on scanning network model or topological model [33] [34], depends on the target network infrastructure. Advantage of using these malware modeling framework is to control the malware propagation and prevent network to overwhelm from random malware transmission and propagation. There are different other methods to handle the challenge of network flooding by malware or malware propagation. The obvious solution is to pick a range of IP addresses in a large network to target one or more specific nodes. Another practical mechanism for randomizing the malware propagation behaviours and protect a large network from flooding, is to introduce time delay modeling (random time factor) in malware propagation. As to set a random time delay for the propagation of each malware sample.

4.3.2 Malware Encryption Process

Encryption is major weapon used by the malware author against malware analyst to counter the malware payload analysis and identification of malware author's intentions. The malware encryption uses the asymmetric encryption algorithms to encrypt the malicious payload.

The encryption key for asymmetric cryptographic algorithms is derived from the environmental variables of target system. Before the malware attack process begin, need to collect the environmental information which can be useful to generate the encryption keys. The environmental variables is combination of data gathered from target system, include IP address, MAC address, system variables, operating system unique parameters, path variables and other network triggers etc. The reason behind the use of environmental keys is to auto decrypt the malware payload on specific target node.

The malware encryption process consist of malware payload and cleartext loader. The malware payload consist of malicious code that need to execute on target system to compromise the opponent node(s). The loader program scans and check the target system for environmental variables and define the way to transforms these variables to generate encryption keys, instead of storing keys insides the malware payload. To generate standards keys from environmental variables the cryptographic one way functions known as hash functions can be used. The hash function transforms the arbitrary length

environmental variables into a fixed length key. For example, the MD5 hash function transforms the arbitrary length input message into 128 bit message digest. Later on, these encryption keys will be used to encrypt the malicious code. The cleartext loader security depends on obfuscation scheme used by malware author.

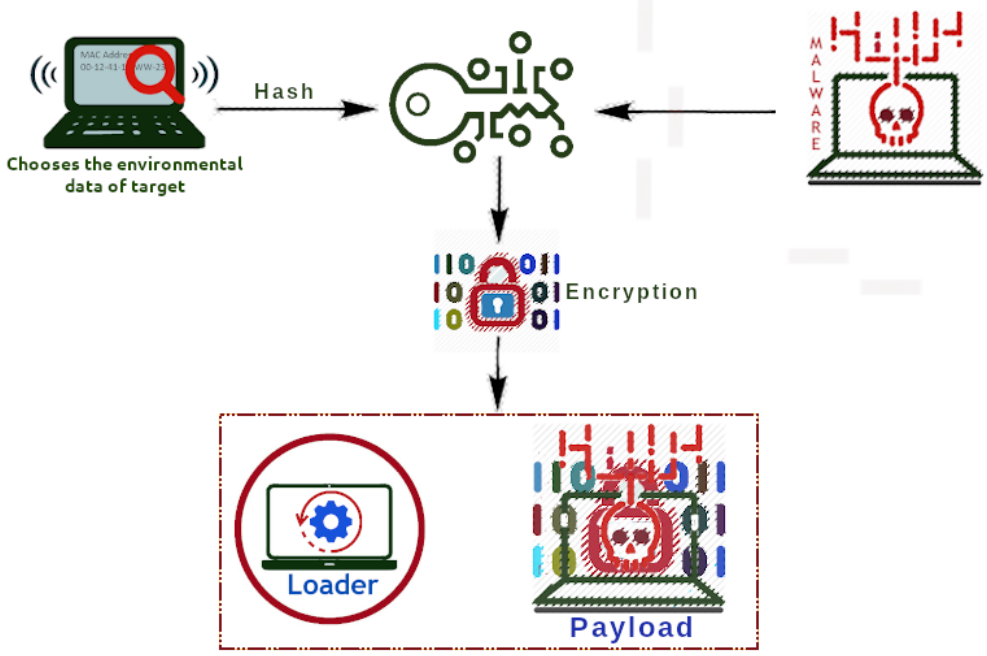

Figure 4.2: Malware encryption process

On the target side, once malicious software infects a new system, the malware loader scans the compromised node's environmental variables, hashes these variables to generate keys, and tries to decrypt the malicious payload through the derived keys. If the malware decryption succeed, the malware payload will execute. If not, copies of the malware will be generated to target successive system.

The malware author initiate with N distinct encrypted sample of same malicious code to infect X initial devices. The analyst goal is to collect these initial X malware sample. Analyst primary objective is to ensure that neither of these X initial variants compromises his devices in the network. To guarantee this, analyst approximately need K decryption for each of his N nodes. Hence, the malware analyst workload to analyse the malware sample is almost XNK.

4.3.3 Malware Re-randomization Process

The malware re-randomization process produces several indistinguishable variants of a malware, rather than copying identical samples. Malware re-randomization can be done using Asymmetric cryptography. The homomorphic property of different public key encryption schemes can be used to re-encrypt the cipher text. The malware re-encryption scheme is based on universal re-encryption scheme [6] that uses the asymmetric encryption algorithm ELGamal to re-encrypt the cipher text.

The re-randomization algorithm inputs includes, a encrypted payload to generate the exact malicious code, with some randomly chosen values in order to create a new cipher text against same plain text. Hence, homomorphic encryption property of ElGamal is used to generate different looking sample of same malware to infects X target nodes without any knowledge of private key.

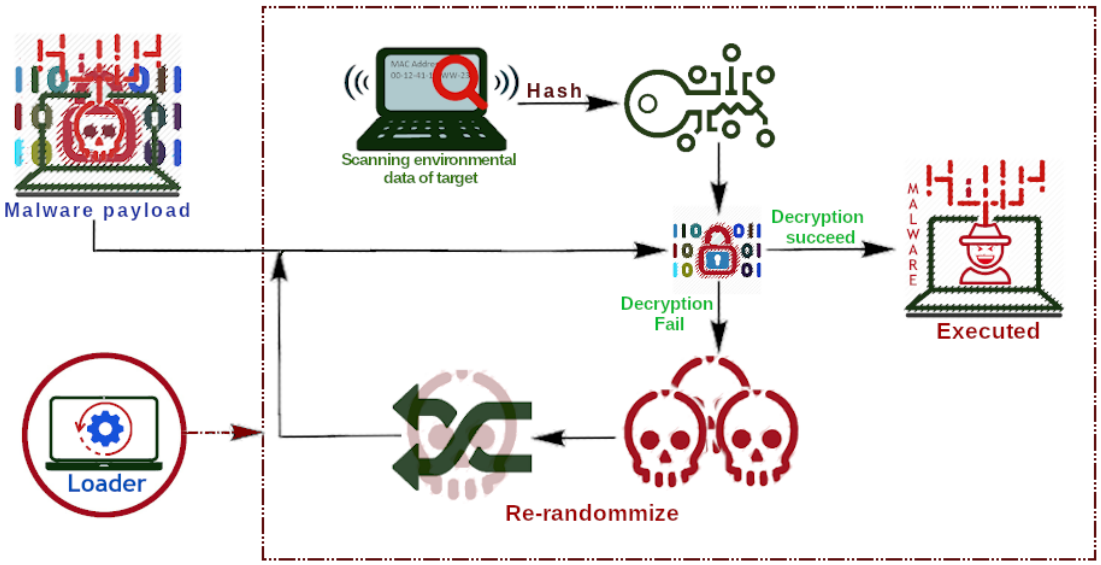

Figure 4.3: Malware re-randomization process

The re-randomization process begins with scanning the target environmental data. After collecting required environmental data, hashes the data to generate the decryption key, to check the decryption procedure. These decryption key(s) is used to decrypt the encrypted malicious payload and checks for decryption succeed or fail. We can look for malware sample signature to verify the decryption results. In windows, the exe file format start with signature 0x4D5A (MZ). Hence, if the malware is windows based,

we will check the .exe file signature to verify the decryption. Otherwise, if malware is Linux based, then the underlying malware will belong to deb or bin file format. The deb file format signature is 0x213C617263683E (!<arch>.) and binary file signature is 0x7F454C46 (.ELF). If the malware decrypted successfully on a node, it will execute. Otherwise, the malware decryption fails. In this case, the re-randomization algorithm takes uniformly random values (ElGamal random factor) as input to re-encrypt the malicious payload to generate the indistinguishable sample of same malware without the knowledge of secret key. The re-encryption uses the homomorphism of ElGamal that allow re-encryption of malicious payload.

4.4 Proposed Implementation of Existing ElGamal Scheme

The malware encryption and re-randomization scheme built on public key cryptosystem ElGamal over group G of primitive order p, with generator g. The framework of malware encryption and re-randomization is same as Galteland extended scheme [10]. The implemented scheme consist of four algorithms, the malware encryption algorithm Enc_m, the malware decryption algorithm Dec_m, the malware re-randomization algorithm $Re\text{-}rand_m$ and malware re-randomized payload decryption algorithm $Re\text{-}rand\text{-}Dec_m$. The decryption algorithm Dec_m will be executed, if the malicious payload is only encrypted (not re-randomized), otherwise (in case of malware re-randomization) the malware re-randomized payload decryption algorithm $Re\text{-}rand\text{-}Dec_m$ will be executed.

4.4.1 Malware Encryption Algorithm

The malware encryption algorithm Enc_m take input a executable file (malware) and transforms it into bit string (bit stream) to encrypt the files with large size. The bit string is than padded with one 1 bit and length 0's bits. The number of zeros padded which is $l * (n + 1) + 1$. Where l is the message length, n is number of re-randomization that will perform on encrypted malicious payload. The padded bit stream length is denoted as L. So the plain text (padded malicious code) m_{L_m} is bit string of length L_m.

4.4.1.1 Encryption parameters

The malware encryption uses the standard ElGamal's parameters. Let G represent the underlying group for ElGamal asymmetric cryptosystem with p as order of group G and generator g. The p represents the big prime number and the generator g is chosen at random number, such that $g < p$ and $gcd(p, g) = 1$. The random factor r and s have chosen as random such that $r, s \in (1, ...(p-1))$.

4.4.1.2 Key generation

To encrypt the malware payload plain text m is encrypted using encryption key k, such that $k \in Z_p^*$. The private key k is generated from environmental data of the target node. In our implementation, we use the MAC address of target system and transforms it into the key k. The MAC address is 48 bit. The key k must be less than prime p. Hence, key k is calculated as $k = mac \mod p$.

4.4.1.3 Encryption factor γ

The encryption factor γ is chosen as random. For encryption process the length of encryption factor γ must be equal to L_m, where L_m is the length of padded malicious code plain text m of padded bit stream length L. For every, variants the value γ will change as it is chosen at random and also the length L_m will vary each time. So we need to define a way to transforms a random encryption factor γ into a string γ_{L_m} of arbitrary custom length L_m.

To transforms the encryption factor γ into to a string γ_{L_m}, we have used the two standard one way hash functions. One is MD5 hash algorithm and the other is SHAKE hash algorithm.

MD5 The MD5 [35] was designed by Ronald Rivest in 1991, is a standard one way hash function that take an arbitrary length input and return 128 bit message digest. The message digest of MD5 usually represented in a hexadecimal sequence of 32 digits. In our implementation, we have used the OpenSSL C library for MD5 to transforms the encryption factor γ into 16 byte string.

SHAKE-128 The SHAKE algorithm [36] belongs to SHA-3 family of XOF(Extendable output functions) algorithms. The extendable-output function(XOF) is a hash function

in which the message digest can be extended to any arbitrary custom length. The SHAKE-128 and SHAKE-256 are two standard SHA-3 XOF's. The suffix "128" and "256" specify the strength of the algorithm instead of indicating the message digest length as in other standard hash algorithms. The SHAKE-128 and SHAKE-256 were the first XOFs (extended output functions) to be standardized by NIST.

In our implementation of malware encryption algorithm, we have input the 16 byte MD5 to SHAKE-128 algorithm and get our required message digest as string γ_{L_m} of arbitrary length L_m. We have used the Odzhan C implementation of SHAKE-128 [37] in our implementation. Hence, the SHAKE-128 algorithm is useful to get the desired length output in the encryption process.

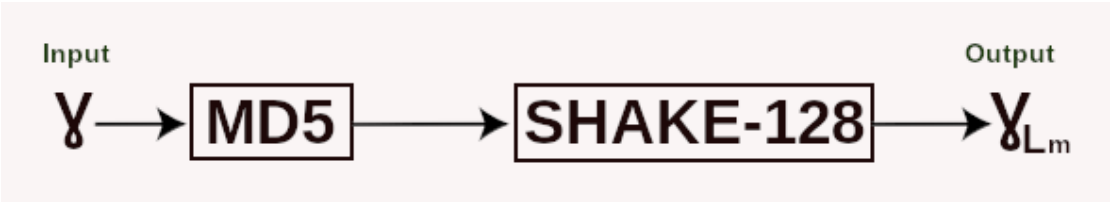

Figure 4.4: Calculation of Encryption factor γ using MD5 and SHAKE-128

4.4.1.4 Encryption

For encryption function, input comprises, padded malicious payload (plain text) m_{L_m}, the key k, the random factor r, s and the encryption factor γ, γ_{L_m}. The output is a cipher text C, which is calculated as:

$$C = c_1||c_2||c_3||c_4||c_5$$

Where c_1, c_2, c_3, c_4 and c_5 is calculated as:

$$c_1 = g^r \mod p$$
$$c_2 = g^{kr} \mod p$$
$$c_3 = g^s \mod p$$
$$c_4 = g^{ks}.\gamma \mod p$$
$$c_5 = \gamma_{L_m} \oplus m_{L_m}$$

4.4.2 Malware Decryption Algorithm

The decryption algorithm Dec_m takes input the cipher text C. The decryption process is done on the target node, so need to auto-decrypt the malicious payload on the target machine using the secret key k. As the key k is derived from target MAC address of the target node. We have extracted the target MAC address and calculate the key k as, $k = mac \mod p$. To decrypt the encrypted malicious payload, required to verify the target node. Hence, we have checked $c_1^k = c2$, where $c_1 = g^r$ and $c_2 = g^{kr}$. In our implemenation, if the verification fails, the loader program force to moves onto re-randomization algorithm to target subsequent node, otherwise, the payload will decrypted and executed.

To decrypt the malicious payload, the first step is to extract the γ from the cipher text, as:

$$\begin{aligned} c_3^{-k}.c_4 &\implies g^{-ks}.g^{ks} \mod p \\ &\implies \gamma \end{aligned}$$

After determining γ, γ_{L_m} will be evaluated using the hash algorithms MD5 and SHAKE-128. Input γ to MD5 hash function, which return 128 bits hash value. The length of cipher text object c_5 is equal to the padded plain text length L_m. The length L_m and 16 bytes message digest of MD5 is input to SHAKE-128 to get γ_{L_m} of length L_m.

To decrypt the padded plain text m_{L_m}, use the cipher text object c_5 and γ_{L_m} , as:

$$m_{L_m} = c_5 \oplus \gamma_{L_m}$$

To determine the padded bit stream length, the implemented loader program search for the first '1' from the end of bit stream of γ_{L_m} and discard the padded bit from the γ_{L_m}, to get the original plain text. As in encryption algorithm, the plain text is padded with tail of zeros and just one 1. To execute the malware payload, loader write the decrypted data to a file. Hence, we get the original malicious payload plain text m. Before executing the malware, loader program will verify the malicious payload using the standard signature of underlying file format. On successful verification, malware payload will be executed.

4.4.3 Malware Re-randomization Algorithm

The re-randomization algorithm Rr-rand$_m$ takes input the cipher text C and the public parameter p. Decryption process is done on the node which prompts the decryption fails or invalid private key for decryption. In consequence, before target the new node, re-encrypt the cipher text without knowledge of private key k. The re-randomization algorithm choses two new random factors r' and s'. The encryption factors γ' to re-encrypt the cipher text C, is also chosen at random. The r', s' and γ' should be less than the public parameter p.

The value of β_{L_β} with length L_β is evaluated as:

$$\beta_{L_\beta} = c_3||c_4||c_5$$

Where the cipher text C objects, $c_3 = g^s$, $c_4 = g^{ks}.\gamma$ and $c_5 = \gamma_{L_m} \oplus m_{L_m}$.
The γ'_{L_β} will be computed by using the hash algorithms MD5 and SHAKE128, as:

$$\gamma' \rightarrow MD5 \xrightarrow{16bytes} SHAKE128 \xrightarrow{L_\beta} \gamma'_{L_\beta}$$

The output of re-randomization algorithm Re-rand$_m$ is cipher text C', which is calculated as:

$$C' = c_1\prime||c_2\prime||c_3\prime||c_4\prime||c_5\prime$$

Where $c_1\prime$, $c_2\prime$, $c_3\prime$ and $c_4\prime$ is calculated as:

$$c_1\prime = c_1^{r'} \mod p$$
$$c_2\prime = c_2^{r'} \mod p$$
$$c_3\prime = c_1^{s'} \mod p$$
$$c_4\prime = c_2^{s'}.\gamma' \mod p$$
$$c_5\prime = \gamma'_{L_\beta} \oplus \beta_{L_\beta}$$

4.4.4 Re-randomized Malicious Payload Decryption

The re-randomized cipher text C' decryption algorithm *Re-rand-Dec*$_m$, is modified form of decryption algorithm Dec_m with some addition. The re-randomize decryption algorithm takes input the public parameters p and g and the cipher text C'. The cipher

text C and C', treated as same way by decryption algorithm Dec_m. The secret key k is generated from the target MAC address to decrypt the payload. To verify the target node for decryption of payload we check $c_1^k = c_2$. Here, the cipher text C instance c_1 is c_1' and c_2 is c_2'.

$$\begin{aligned} c_1^k &= c_2 \\ c_1^{r'k} &= c_2^{r'} \\ g^{rr'k} &= g^{krr'} \end{aligned}$$

On verification fails loader program encforce to moves onto re-randomization algorithm to target another node, otherwise, the payload will decrypted and executed. To decrypt the payload, the first step (additional step in decryption algorithm) is to determine γ' from cipher text C', as:

$$\begin{aligned} c_3'^{-k}.c_4' &= c_1^{s'(-k)}.c_2^{s'}\gamma' \\ &= g^{rs'(-k)}.g^{krs'}\gamma' \\ &= \gamma' \end{aligned}$$

After that, the loader program computes the γ'_{L_β} using the hash algorithms MD5 and SHAKE128. The length L_β is same as the length of c_5'. The γ'_{L_β} as;

$$\gamma' \rightarrow MD5 \xrightarrow{16bytes} SHAKE128 \xrightarrow{L_\beta} \gamma'_{L_\beta}$$

By using the cipher text c_5' and γ'_{L_β}, we can find β_{L_β}, as:

$$\beta_{L_\beta} = c_5' \oplus \gamma'_{L_\beta}$$

The β_{L_β} is concatenation of cipher text's C instances, as, $c_3 \mid\mid c_4 \mid\mid c_5$. In other words, $\beta_{L_\beta} \implies g^s \mid\mid g^{ks}.\gamma \mid\mid \gamma_{L_m} \oplus m_{L_m}$. The loader will determine the length of g^s and $g^k s.\gamma$, as they have equal in length with cipher text C' instances c_3' and c_4' respectively. Hence, we can find c_3, c_4 and c_5 from β_{L_β}. This procedure continue for n iterations, where n is the number of re-randomization performed on encrypted payload.

Next step is to computer γ, from the cipher text C using the secret key k, as:

$$\begin{aligned} c_3^{-k}.c_4 &= g^{-ks}.g^{ks} \mod p \\ &= \gamma \end{aligned}$$

Now, our implemented loader will compute γ_{L_m} from γ and c_5's length L_m, as:

$$\gamma \rightarrow MD5 \xrightarrow{16bytes} SHAKE128 \xrightarrow{L_m} \gamma_{L_m}$$

The final step is to determine the malicious payload's padded plain text m_{L_m}, as:

$$m_{L_m} = c_5 \oplus \gamma_{L_m}$$

The padded bit stream length will be determine by searching the first '1' from the end of bit stream(right to left) of γ_{L_m} and discard the padded bit from the γ_{L_m} to get the original plain text m. As in encryption algorithm, the plain text is padded with tail of zeros and exactly one 1. The loader than writes the decrypted data to a file to execute and launch the malware. The re-randomization decryption algorithm need to execute n time on the encrypted malicious payload. Where n is the number of re-randomization performed on the malicious payload. Before execution of the malware, we have used the standard executable file format signature to validate the malicious payload. Malicious software payload will be executed on successful validation.

4.5 Results

In this section, we present experimental setup and experimental results of our propsed implementation of ElGamal based malware encryption and re-randomization scheme.

4.5.1 Experimental Setup

The ElGamal malware encryption and re-randomization scheme algorithms have implemented using the C language (KDevelop Platform Version; 5.2.1) on Ubunut 18.04(Linux). Experiments are performed on Intel(R) Xeon(R) CPU E5-1660 v3 with a 3.00 GHz and 16 GB of memory. We used some popular malware samples in our experiments, as show in table 4.1.

Malware	Platform	Signature	Size
WannaCry[38]	Windows	MZ	3.5 (MB)
Zeus[39]	Windows	MZ	252.9(KB)
GandCrab[40]	Windows	MZ	124.4(KB)
Kovtreer[41]	Windows	MZ	431.9 (KB)
Wirenet[42]	Linux	.ELF	64.4(KB)
Encoder[43]	Linux	.ELF	317.5(KB)
Dendroid[44]	Android	PK	942.8 (KB)

Table 4.1: Malware Samples

4.5.2 Experimental Results

We evaluated the performance of our proposed implementation of malware encryption, re-randomization and decryption algotiyhms by using parameters *Enc(sec)*, *Dec(sec)*, *Re-rand(sec)* and *Re-randDec(sec)*. All these algorithms generate different time (T1, T2, T3 and T4) accourding to variations in algorithms paramenters. Table 4.2 shows the experimental results against different malware sample.

Malware	Enc (sec)				Dec (sec)				Re-rand (sec)				Re-randDec (sec)			
	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4	T1	T2	T3	T4
WannaCry[38]	2.392	2.384	2.356	2.356	7.982	16.6	2.532	13.934	1.540	1.532	1.534	1.521	9.252	22.485	9.824	23.430
Zeus[39]	0.171	0.162	1.164	0.17	4.572	1.001	9.98	0.721	0.105	0.117	0.113	0.111	8.837	14.454	15.172	14.175
GandCrab[40]	0.086	0.084	0.085	0.081	3.304	5.01	12.114	2.051	0.055	0.05	0.055	0.051	11.92	11.844	15.737	12.458
Koveter[41]	0.003	0.002	0.002	0.003	4.163	15.69	7.354	16.696	0.002	0.001	0.002	0.002	8.813	31.036	20.678	20.012
Wirenet[42]	0.041	0.039	0.043	0.044	2.726	8.657	2.798	12.271	0.028	0.028	0.028	0.028	4.971	24.674	7.753	29.429
Encoder[43]	0.209	0.208	0.207	0.21	15.416	13.982	16.68	3.579	0.137	0.134	0.137	0.137	24.061	21.096	16.177	7.815
Dendroid[44]	0.637	0.634	0.643	0.629	8.406	0.538	6.057	1.826	0.407	0.412	0.409	0.402	16.188	2.459	17.39	13.605

Table 4.2: Performace analysis of ElGamal based scheme

Chapter 5

Paillier- Malware Encryption and Re-randomization Scheme

5.1 Introduction

In this chapter, we will propose a new scheme for malware payload encryption and re-randomization, based on asymmetric cryptosystem, Paillier. First, we introduce the overall malware prospective and Paillier algorithm's encryption and decryption process with homomorphic property. Further, we describe the use of PHE scheme for malware re-randomization, probabilistic and deterministic encryption. We also explain the concept of semantic security and limitation of RSA cryptosystem for malware re-randomization. At last, we present our proposed scheme for malware encryption and re-randomization, based on Paillier's semantically secure cryptosystem and implementation results.

5.2 Malware Encryption

As the development and propagation of malware improves, methods used to conceal malicious activities by malware developer, have increased. Attackers have invested a lot more work into rendering the process extremely complicated and time intensive to detect malware. Growing numbers of anti-sandbox, anti-debug, and anti-analysis methods(such as dead code, cryptography encryption, etc) hinder application's static and dynamic malware analysis and increasing uncertainties in malware analysis. In tandem with software and innovative technology, the malware architecture has evolved,

evolving modern industry methodologies and approaches to their malware frameworks and exploiting new services as they release attractive options for adversaries.

The underlying cat and mouse challenge which has existed for decades among malware writers and anti-malware business firms or malware analysts. There is a significant competition among malware analysts and malware developers. Each opposing party is trying to expand their ability to overcome the opponent. Among the other critical concerns on the perspective of malware for malicious software author is to extend the malware's lifespan in the open ocean, as much as achievable. Our goal is to propagate malware, prevent propagating malware from analysis and to hide the malware developer intentions. We will propose a novel scheme based of Paillier cryptosystem, to encrypt and re-randmomize the malware payload. The proposed scheme is more efficient than ELGamal based malware encryption and re-randomization scheme, earlier described in Chapter 4.

5.3 Paillier Cryptosystem

The Paillier crypto-system is a probabilistic asymmetric cryptosystem is based on composite residuosity problem[23], proposed in 1999 by Pascal Paillier [19]. It is belonged to the family of RSA and ElGamal of PHE scheme. It is a probabilistic asymmetric algorithm that inherits the additives homomorphic properties for public-key cryptography. The Paillier cryptosystem's algorithms are:

5.3.1 Paillier Key Generation

The Paillier's key generation algorithm $KeyGen$ comprises two big prime integers p and q of equal length, chosen at random as a input such that the $p.q$ and $(p-1)(q-1)$ are relatively prime to each other as:

$$gcd(p.q,\ (p-1)(q-1)) = 1$$

Subsequently, the plain text sender computes the value n as, $n = pq$ and λ as:

$$\lambda = lcm(p-1,\ q-1)$$

Yet again, randomly choose an integer g following $g \in Z^*_{n^2}$ such that:

$$gcd(n,\ L(g^{\lambda} \mod n^2)) = 1$$

$$Here, L(x) = (x-1)/n,\ \forall x \in Z^*_{n^2}$$

As a result, we get the public key pair (n, g) and private key pair (p, q) as Paillier $KeyGen$ output.

5.3.2 Paillier Encryption

The Paillier encryption algorithm Enc takes a message as input m, where $m \in Z_n$ and randomly pick an integer r, where $r \in Z^*_n$, this random element is being used to fulfil the probabilistic algorithm property that a specific plaintext could have multiple ciphertexts. This random parameter does not affect the accurate decryption but modifies the ciphertext.

The output result of encryption algorithm Enc is ciphertext c, where $c \in Z_{n^2}$ and can be generated as:

$$c = g^m.r^n (\text{mod } n^2)$$

5.3.3 Paillier Decryption

The user can decrypt the ciphertext c using the private key λ in decryption algorithm Dec, to get the original plaintext message m, as:

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)}$$

Just because of the property that n and $L(g^\lambda \mod n^2)$ are relatively prime to each other, it is feasible to evaluate the inverse of $L(g^\lambda \mod n^2) \mod n$.

5.3.4 Paillier Homomorphic Property

The Paillier asymmetric cryptosystem is a PHE scheme with homomorphic additive property. This means addition operations can be performed with ciphertext without losing or manipulating the original data.

For input plain text messages m and m' where $m, m' \in Z_n$,

$$\begin{aligned} Enc(m) * Enc(m') &= (g^m.r_1^n \mod n^2) * (g^{m'}.r_2^n \mod n^2), \\ &= (g^{m+m'}.(r_1 * r_2)^n \mod n^2), \\ &= Enc(m + m'). \end{aligned}$$

This implies that the encryption of two plaintext messages m and m' is precisely the multiplication of the corresponding ciphertext c and c' as above mentioned, Paillier crypto-system is homomorphic over addition. But it also support some additional operation over plain text m, m' as:

$$Enc(m) * Enc(m') \ (\mathrm{mod} n^2) = Enc(m + m') \ (\mathrm{mod} n)$$
$$Enc(m) * g^{m'} \ (\mathrm{mod} n^2) = Enc(m + m') \ (\mathrm{mod} n)$$
$$Enc(m)^k \ (\mathrm{mod} n^2) = Enc(km) \ (\mathrm{mod} n)$$

5.4 PHE Schemes for Malware Re-randomization

Partially homomorphic encryption (PHE)[45] helps to keep confidential information protected by enabling only specific mathematical operations on encrypted data to be performed. This implies that an infinite number of occasions a single operation could be conducted on the encrypted data. RSA public key cryptosystem is a PHE scheme, which is commonly used to establish secure connections through SSL / TLS. Some other examples of PHE scheme are ElGamal public key cryptosystem (a multiplicative scheme) and Paillier cryptosystem (an additive scheme). Our goal is to use the PHE scheme to re-encrypt the malicious payload. In chapter 4, we have discussed the use of ElGamal cryptosystem for malware encryption and re-randomization scheme. Here, we explain probabilistic and Deterministic encryption, the concept of semantic security and the limitation of RSA asymmetric PHE scheme for malware encryption and re-randomization. Further, we justify our proposed scheme which is based on Paillier cryptosystem to encrypt and re-randomize the malicious payload.

5.4.1 Probabilistic and Deterministic Encryption

The deterministic encryption is a type of encryption that generates the same output or encrypted data, against the same input plain text, even for different executions. The typical examples of deterministic cryptosystem are unpadded RSA and several block ciphers when using electronic code book mode or same IV.

On the other side, there are more than one possible ciphertext for each plaintext in a probabilistic encryption scheme. The ElGamal and Paillier public key encryption are examples of probabilistic encryption algorithms.

5.4.2 Semantic Security

Semantic security is a notion of defining a cryptographic encryption algorithm's security. An opponent is allowed to pick one of two plaintext, m and $m\prime$ and gets the one of the ciphertext of m or $m\prime$. If an opponent can not predict with a probability more than half(1/2) that the received ciphertext is the encryption of either the plain text m or $m\prime$, the encryption algorithm is considered semantically secure, sometime refers to as encryption indistinguishability.

Goldwasser and Micali initially described it and published it in their 1984 influential article "Probabilistic Encryption" [46]. Typically, this feature is obtained by adding a random factor into the cryptogrpahic encryption algorithm. A quite simplest way of transforming any deterministic encryption scheme into some kind of probabilistic algorithm is to pad the plain text before encrypting it with a random sequence and unpad it only it after the decryption. In the block ciphers, it can be easily achieved using the modes of encryption (Cipher Feedback, Cipher Block Chaining etc), while in public key encryption scheme, each time, by choosing the random factor r, before encrypting the plain text, the corresponding ciphertext will alter each time against same input plaintext. In block cipher this can be achieved by using initial vector(IV) or modes of encryption. If there were the identical ciphertext for each message then the encryption algorithm will be deterministic and it would neither semantically secure and nor indistinguishable. The examples of semantically secure encryption scheme are Goldwasser-Micali [46], ElGamal and Paillier [19].

5.4.3 RSA vs Paillier for Malware Re-randomization

The RSA and Paillier cryptosystem, both belongs to partially homomorphic encryption (PHE) scheme family, with multiplicative homomorphism and additive homomorphism respectively. The Paillier asymmetric algorithm has the featured characteristic that a plaintext has many different cipher-texts making it more resilient against a wide variety of different cryptographic attacks (semantically secure). This is because the encryption is injected with the inherent randomness, which can be eliminated by the key because of its quadratic residuosity property. To encrypt a message m, the Paillier encryption algorithm uses and pick the random r, where $r \in z^*_{n^2}$, and compute the ciphertext as: $c = g^m.r^n \mod n^2$. Hence, every message m may have several equally legitimate

ciphertexts dependent on the value or size of n and the divisibility relationships between some of these values.

In our proposed scheme, we exploit the Paillier homomorphic property and its probabilistic property (the random factor r), and use it to encrypt the malware and then re-randomize the encrypted malicious payload without revealing the corresponding plain text or the private key.

RSA support the multiplicative homomorphic encryption, already explained in section 2.4.2.1. For malware re-randomization, the requirement is to re-encrypt the encrypted payload or ciphertext without knowledge of secret key. The both ElGamal and Paillier's random factor r helps us to re-encrypt the encrypted malicious payload without revealing the plain text. In case of RSA, by default the RSA is not semantically secure, as it is deterministic encryption scheme. Although, RSA has homomorphic encryption but is not sufficient to re-encrypt the ciphertext because there is no random factor, that helps us to re-randomize the cipher text as in ELGamal and Paillier.

RSA can be either semantically secure or homomorphic, not both. In practice, before encrypting the plain text message m, RSA can add the randomness (e.g. RSAES-OAEP) [46], which provide semanatic security, but completely looses the homomorphic property.

Some other approaches have been used by researchers, to use RSA for re-encryption of ciphertext in proxy re-encryption. Most of the schemes [47], split the algorithm into two parts. The first one is, the algorithm will use the original private key as well as the fresh public key and create a kind of intermediate key. The second part is the key will then be circulated to untrusted entities which use the intermediate key and the new public key to update their encrypted data to the new key pair. But in our case, we need to auto-decrypt the malicious payload on the target node by using environmental keys. It is impractical to generate the new key pair to re-encrypt the ciphertext. The reason is, the private key is not appended with malware payload or in text loader. It will be generated on run time from target node. Hence, it not feasible to use RSA cryptosystem to re-randomize the malware payload.

5.5 Design and Implementation of Proposed Scheme

Our proposed scheme is based on the probabilistic public key encryption algorithm, Paillier. Our goal is to prevent malware payload from analysis by increasing the malware analyst's workload. Encrypting malware payload obstruct to identify malware author's intentions. The malware re-randomization or re-encryption aid to hide the identity of malware author and make each malware sample indistinguishable. We proposed a framework for malware encryption and re-randomization of malicious payload on the target node without revealing the plaintext or the private key. The secret key is generated from the target environmental data, to encrypt the malicious payload using Paillier encryption algorithm. To execute malicious payload on the target node, need to extract the environmental key and use it to decrypt the malware payload. On successful decryption the malicious payload will execute. Otherwise, re-encrypt the encrypted payload, and transmit it to next node to find the target and execute. The Proposed scheme follow the malware propagation model, describe in Section 4.3 of Chapter 4. We have implemented our proposed scheme based on Paillier for malware encryption and re-randomization, using C language on Linux plateform. For big integer value(greater than 8 bytes), we have used the GMP C library. Our proposed scheme include three main algorithms, malware encryption algorithm, malware re-randomization algorithm and the malware decryption algorithm. The malware encryption and decryption algorithms have sub algorithms, the encryption key generation and decryption key generation algorithms, respectively.

5.5.1 Malware Encryption

The malware encryption process aids the malware writers by encrypting the malicious payload to maximize the workload for analysis. Encrypting the payload restricts the malicious software from being reversed by an analyst and obscures the malware author's desires. Environmental data is collected from the target network and therefore could consist of, for instance, IP address, directory paths, PATH variables etc. In our work, we use MAC address as environmental variable to generate encryption and decryption keys. The mechanism for malware encryption consist of the payload and the cleartext loader for malware. The malware payload consists of malicious code to be run on target system in order to access the node(s) of the challenger. The loader software scans and tests environmental variables on the target system and defines how these variables can

be converted to produce encryption or decryption keys rather than keeping keys within the payload of malware. The encryption process consist of the private(encryption) key generation algorithm and encryption algorithm.

5.5.1.1 Encryption Key Generation Algorithm

The encryption key generation algorithm $KeyGen_{Enc}$ is based on Paillier key generation, which takes two prime numbers p and q (of equal size) as a input, where p and q are relatively prime to each other. On the other side, the decryption or secret key λ depends on p and q, and can be computed as, $\lambda = lcm(p-1, q-1)$.

But, in our case it is impractical. Because, for malware propagation, our requirement is to auto-decrypt the malicious payload on the target system. It can be achieved using the key(s) generated from taget environmental data. This implies that the decryption key λ must be depended on the target environmental data, and can be re-generated.

To overcome this problem, we generate the prime p and q, from target environmental data (MAC address of target system), and computes a modulus $n = p * q$. Now, choose a random number $g \in Z^*_{n^2}$. The order of g is multiple of n. Selecting $g = n + 1$, is effective choice and can be easily computed [48].
The output of Encryption Key Generation algorithm $KeyGen_{Enc}$ is public (encryption) key pair (g, n).

5.5.1.2 Encryption Algorithm

The encryption algorithm Enc_m takes the malware executable file as input and transform it into array of string, as plain text message m. The encryption algorithm Enc_m uses the public key pair (g, n), generated from key generation algorithm $KeyGen_{Enc}$ to encrypt the plain text. Select a Paillier's random factor r, where $r \in Z^*_{n^2}$.

But, the Paillier cryptosystem allows encrypting integers modulo n. Therefore, if input plain text message m is bigger than n, encrypting it will lose most of the plain text message m, only $m \mod n$ is retrieved through decryption. In case of malware encryption, it is likely that $m>n$.

To encrypt a message bigger than n, we break it into blocks, which encrypt separately.

We can write m in base n, as:

$$m = \sum_{i=1}^{c} m_i$$

Where c is number of chunks or blocks and each $m_i < n$.

Encrypt the m_i's separately using the public key pair (g, n) and random factor r. The cipher text c instances c_i's computed as:

$$c_i = g^{m_i} * r^n \mod n^2$$

5.5.2 Malware Re-randomization

Re-randomizing the malicious payload, generates several indistinguishable variants of a malicious software rather than just replicating the same samples. Re-randomization of malware can be achieved using public key cryptographic algorithm's homomorphism. The inputs of the re-randomization algorithm include an encrypted payload, to create the exact malicious payload (encrypted), with certain randomly selected values to create the new cipher text against the encrypted payload of same plain text. Paillier's homomorphism and probabilistic property (random factor) is used to create different looking samples of the same malware.

The re-randomization algorithm Re-rand$_m$ takes input the cipher text c_i, the public parameter n. Choose the random factor $r' \in Z^*_{n^2}$ and re-randomize the encrypted malicious payload (cipher text) c_i's. The re-randomization algorithm's output c'_i is:

$$c'_i = c_i * r'^n \mod n^2$$

5.5.3 Malware Decryption

The malware decryption process begins with scanning the target environmental data. After collecting required environmental data, generates the private key from the acquired data, and decrypt the malicious payload using the obtained private key. The decryption process consist of two algorithms, the private key generation and the decryption of ciphertext.

5.5.3.1 Decryption Key Generation

The decryption key to decrypt and execute the encrypted or re-randomized malware payload, is generated form target environmental data. The first step is to determine the target MAC address (environmental variable), and generate the prime p and q from MAC address (6 bytes). After that the private key λ, will then generated from p and q, as:

$$\lambda = LCM((p-1)(q-1))$$

5.5.3.2 Decryption Algorithm

The decryption procedure is same for both encrypted (only) and re-randomize (re-encrypted) payload, so $c' = c$. The decryption algorithm takes input the cipher text instances c_i, the private key λ and public key parameter n. The Decryption algorithm outputs the plain text message instances m_i, as:

$$m_i = \frac{L(c_i^{\lambda} \mod n^2)}{L(g^{\lambda} \mod n^2)}$$

where, $L(x) = (x-1)/n$, $\forall x \in Z_{n^2}^*$.

After decryption of cipher text instances c_i's, we get the plain text instances m_i's. We combine all the m_i's to get the plain text message m and write the result in file to verify and execute the malware payload.

In our implementation, the loader program checks for signature of malicious software samples to validate the results of decryption. For windows, the executable (exe) file format starts with 0x4D5A (MZ) signature. So, the text loader program will check the .exe file signature to validate the decryption for windows based malware. Otherwise, the underlying malware will belong to the deb or bin file format if malware is based on Linux. The signature of deb file format is 0x213C617263683E! (< arch>.) and 0x7F454C46 (.ELF) is the signature of the binary executable file. On successful verification of malware payload, the loader program execute the malicious payload.

5.6 Results

In this section, we present experimental setup and expiremental results of our propsed paillier based malware encryption and re-randomization scheme.

5.6.1 Experimental Setup

The Paillier based malware encryption and re-randomization scheme (proposed scheme) algorithms have implemented using the C language (KDevelop Platform Version; 5.2.1) on Ubunut 18.04 (Linux). Experiments are performed on Intel(R) Xeon(R) CPU E5-1660 v3 with a 3.00 GHz and 16 GB of memory. For big integers calculation, we have used the GNU GMP library for C. We have used some popular malware samples in our experiments, as shown in Chapter 4, Table 4.1.

5.6.2 Experimental Results

We evaluated the performance of our proposed scheme's implementation of malware encryption, re-randomization and decryption algorithms by using parameters *Enc(sec)*, *Dec(sec)*, *Re-rand(sec)* and *Re-randDec(sec)*. All these algorithms generate different time (T1, T2, T3 and T4) accourding to variations in algorithms paramenters. Table 5.1 shows the experimental results against different malware sample.

Malware	**Enc (sec)**				**Dec (sec)**				**Re-rand (sec)**				**Re-randDec (sec)**			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
WannaCry[38]	6.977	6.968	6.965	6.963	4.152	4.136	4.14	4.141	1.001	1.003	1.004	1.003	4.138	4.166	4.127	4.168
Zeus[39]	0.473	0.479	0.475	0.479	1.384	1.379	1.38	0.192	0.072	0.072	0.072	0.071	1.383	1.374	1.387	1.388
GandCrab[40]	0.232	0.231	0.232	0.23	1.282	1.274	1.272	1.269	0.035	0.034	0.036	0.035	1.276	1.281	1.276	1.267
Koveter[41]	0.819	0.824	0.829	0.836	1.532	1.54	1.531	1.536	0.127	0.123	0.123	0.123	1.533	1.539	1.54	1.534
Wirenet[42]	0.119	0.116	0.118	0.119	1.232	1.225	1.23	1.224	0.018	0.018	0.018	0.018	1.224	1.22	1.218	1.225
Encoder[43]	0.587	0.588	0.59	0.594	1.437	1.434	1.425	1.424	0.091	0.091	0.091	0.089	1.445	1.44	1.438	1.447
Dendroid[44]	1.891	1.884	1.879	1.883	1.965	1.968	1.961	1.949	0.266	0.27	0.269	0.268	1.986	1.972	1.963	1.971

Table 5.1: Performance analysis of Paillier based scheme

Chapter 6

Encrypted Malicious Software Analysis

6.1 Introduction

In this chapter, we provide an overview of methodology used by malware analyst to detect or analyse encrypted malicious software. First we describe the latest malware trends and global malware volume. At last, we describe commonly used malware detection techniques for encrypted malware.

6.2 Malicious Software Analysis

With the growing usage of end-to-end encryption in industrialized structures, the threat of using encrypted networks to spread hidden malware, increasingly becomes a key concern. The malware authors don't want the malware analyst to detect and analyse their malicious software. Malware authors employ anti-reverse engineering methods and anti-analysis strategies such as packing, encryption and other obfuscation methodologies to hide their malicious software. Hundreds of malicious software are launched daily online. Many of these malicious programs use some type of obfuscation from common XOR encryption to even more advanced methods of anti-analysis, and encryption. If a malware is encrypted and packed, there is no likelihood of static analysis. In such situations, the best appropriate approach seems to be dynamic analysis. However, the problem is to figure out how to analyse behaviour to automatically detect malware as

well as how to quantify behaviour. In this chapter, we will explain commonly used techniques for encrypted malware analysis.

6.3 Encrypted Malware Landscape

For the first half of 2019, SonicWall [49] collected data from its monitoring sensors in over 200 nations. The SonicWall detected more than 4.78 billion malware attacks, which is a decline of 20 percent year from first half of 2018. Other types of attacks overall significantly increase during the first half of 2019 as the total volume of global malware declines.

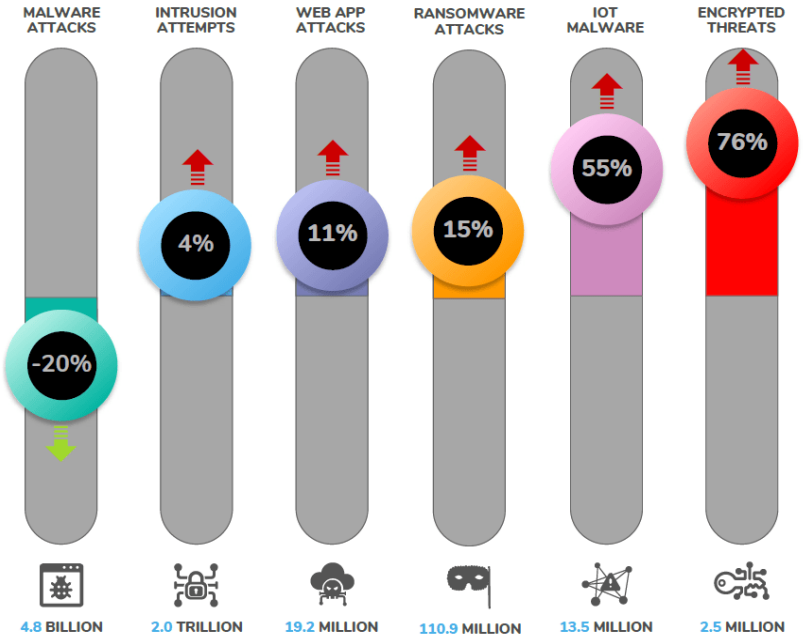

Figure 6.1: Global Malware attack trends 2019

According to Sonywall, the malware authors, also misuse encrypted channels like HTTPS and VPN services based on SSL to mask their data and malicious software. SonicWall have seen around 1,100 attempts of encrypted attacks per customer per day. Most corporations, he says, misleadingly imagine that encrypted communication is the legal traffic. SonicWall [49] has identified 2.4 million attacks which uses encryption, throughout the first six months of 2019, almost surpassing the 2.8 million encrypted threats discovered throughout 2018 (already a 27 percent bounce over the recent year), that represents a rise of 76 percent for initial 6 months. Considering these figures, the total amount of

malicious attempts in the first six months of 2019 has declined by 20 percent to hit 4.78 billion, falling from 5.99 billion in first half of the 2018, the report indicates [49]. In 2018, there had been 10.52 billion attacks reported.

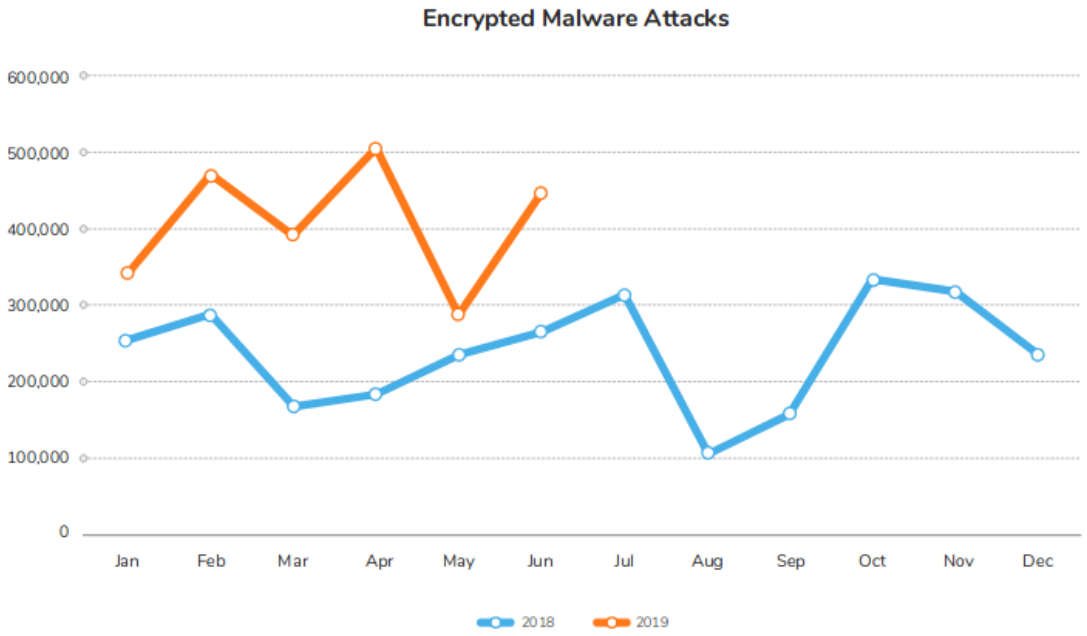

Figure 6.2: Encrypted Malware Attack 2018 vs 2019

6.4 Encrypted Malicious Traffic Analysis

Most enterprises currently don't have an approach in encrypted traffic to uncover malware. The absence of the protection skills, tools and resources to incorporate an approach which can be employed across their communications infrastructure without slowing down the service.

For information security firms(that provide the services for data security by means of encryption), the identification of cyber attacks in encrypted data traffic presents a distinctive type of problems. Monitoring the traffic for cyber attacks and malicious software is essential, but doing so in a way that protects the customer data confidentiality, is another challenge.

6.4.1 Entropy-based Analysis

Lyda and Hamrock introduced the concept of identifying encrypted data through entropy [50]. Because it is reliable and simple to implement, the approach has been commonly used. Many non-encrypted files, however, could have low entropy values, contributing to false positives. For instance, OS Windows XP executable ahui.exe and dfrgntfs.exe with entropy of 6.510 and 6.590 for their .text portion accordingly [51]. Although entropy-based approaches may be successful against obfuscation. encryption or packing, they are unreliable for anti-disassembly techniques, even against basic Bit XOR encryption. Eventually, a file's entropy level could even be intentionally decreased to an entropy level closely related to that of a regular program [52].

6.4.2 Signature-based Detection

PEiD, which deploys about 620 packer and cryptor fingerprints, is a common signature-based platform to identify packed data. Eventually, the entropy rating of a exe could also be intentionally increasing to obtain an entropy value equivalent to that of a conventional system [53]. The downside of such a mechanism however is that only known packers can be identified, and advanced malware typically uses techniques to design a unique encrypted malware or packers. Moreover, if we modify only a signal byte of encrypted malware, the file signature will totally change to bypass from being detected. At last, adding a new encrypted malware fingerprint as it generally requires manual analysis to obtain an accurate fingerprint is a time-consuming task.

6.4.3 File Header-based Analysis

The researcher have done lot of work on file header based anlayiss of malware [54][55][56][57]. But these methods only achieve better performance, when the packer or encrypted malware substantially adjusts the PE header. However, several packers showcase easily recognizable alterations in the PE header of executable file. Nevertheless, customized packers and auto-encryption malware are not necessarily the case. However, if encrypting the malware or packer fully ignores the PE file headers and relies on just the instructions string, there really is no evidence in the executable malicious file header. Finally, even when a encrypted malware binds certain recognizable signature into the header,

plenty of them may be deleted without influencing the credibility of the executable file by the malicious software author.

6.4.4 Hidden Markov Model based Detection

Profile Hidden Markov Models (PHMM) [58], which is known to determine the relationship between deoxyribonucleic acid and protein sequences, may be used to identify malware too. Although PHMM may detect malware such as metamorphic malware, to train them, they need a test data. Therefore, it takes time to process the data, disassemble them, train and scoring the whole dataset.

6.4.5 Machine Learning

Malware identification using machine learning methods has been extremely common in recent years. Machine learning was described by Tom Mitchell as studying computational algorithms which strengthen by experiments [59]. Robert introduced malware identification dependent on statistical behavioral analysis (characteristics). His test findings indicate that the mean identification efficiency reached 90 percent through the use of a classified technique, applicable to only twenty attributes [60]. The benefit of artificial intelligence practices is that it somehow detects recognized malware, as well as acts as understanding for fresh malware detection. Naive Bayes [61], Decision Tree [60], Data Mining [62], Neural Networks [60]., and Hidden Markov Modes [63] seem to be the famous machine learning approaches between many experts to quickly identify 2nd generation malicious software.

6.4.6 Cisco Encrypted Traffic Analytics(ETA)

Cisco announced in June 2017 ETA [64], an analysis product which is able to analyse encrypted traffic in order to identify malware as well as and other traffic without decrypting SSL, and it is used to predict that whether or not abnormalities exists in the network traffic, such as malicious software. ETA utilizes passive monitoring, retrieval of related data components, and supervised global cloud-based artificial intelligence and machine learning. ET-Analytics exports the relevant data components in the form of NetFlow record fields to identify malware in the packet flow, and NetFlow record fields

comprise IDP (initial data packet) and SPLT (Packet Length and Time sequence). According to Cisco Encrypted Traffic Analytic white paper [64], Cisco approached software testing and certification firm Miercom to analyse the efficacy and performance of ETA. The tests were impressive and the Miercom Performance Verified certificate was awarded to Encrypted Traffic Analytics(ETA).

6.5 Conclusion

In this chapter, we provide an overview of existing method for encrypted malicious software analysis. Although there are many studies and research have been conducted on the encrypted malicious traffic analysis, but from a practical perspective , quite few of them can be used for encrypted malware analysis. The entropy based, signature based and file header based analysis techniques have lot of limituation for encrypted traffic analysis. However, the machine learning techiques and cisco ETA framwork is effective and can be used for real time encrypted malicious traffic analysis.

Chapter 7

Conclusion

7.1 Introduction

The chapter is meant to provide a summary of previous chapters so as, to conclude the work. It gives an overview of malware encryption and re-randomization scheme based on ElGamal public key cryptosystem and also highlight our contribution of this work. We also summarized our proposed work, malware encryption and re-randomization based on Paillier cryptosystem. At last, on the basis of our research work, we discuss the future research directions.

7.2 Conclusion

It is important and appropriate to use cryptographic primitives for information security and confidentiality, but what we have seen over the past few years is that malware authors and adversaries are constantly using this encrypted channel to conduct destructive operations across the network.

The rat-race between malware authors and anti-malware innovations has rendered Malware a market by itself worth a billion dollars. While anti-malware tools also developed through the years, the methods of malware authors have often adapted accordingly. Cyber criminals in general utilize cryptography to mask destructive operations, making it harder and harder to identify as more companies turn to encryption to protect data. In order to understand the defence mechanism against it, important to understand the encryption schemes used by malware authors.

In particular, law enforcement agencies can use malware encryption as a tool against terrorists and criminals to expose their crimes by operating in a secure environment. These issues are the driving force behind our research and allowed us to develop a comprehensive system for malware encryption re-randomization and propagation of malware. The goal of our research is to evaluate malware encryption, malware propagation and how malware propagation can be protected from analysis. Our research focus on designing a framework for malware propagation that prevent malware analyst to analyse and reverse engineering the malicious payload.

We have implemented the existing scheme for malware encryption and re-randomization based on ElGamal cryptosystem. Encrypting the malware payload prevent malware from being reversed and hide the malware authors intentions. The malware encryption process uses the MAC address (as environmental key) of the target system, to encrypt the malicious payload so that it can be automatically decrypt at attack time. The re-randomization process uses the homomorphism of ElGamal to re-encrypt the malware payload without any knowledge of plaintext or the private key. The re-randomization process generate multiple in-distinguished malware samples, prevent the malware analyst to reverse engineering toward the source path.

The novel proposed scheme for malware encryption and re-randomization is built on the Paillier cryptostem. The encryption process uses the the numbers prime p and q, generated from same MAC address(as environmental key) of target node. The major difference between ElGamal based scheme and Paillier based scheme, the Elgamal encryption key is directly generated from the target MAC address while the Paillier private key is generated from the two variable p and q, which are derived from the target MAC address. The Paillier based scheme's re-randomization process exploits both the homomorphic and probabilistic property of Paillier cryptosystem to efficiently re-encrypted the malware payload without knowing the actual malware payload(plaintext) or private key.

On the defensive side, to analyse the encrypted malware, different techniques described in the previous chapter with their limitation. These include entropy based detection, file header identification, signature based detection, the hidden markov based malware detection, machine learning and Cisco ETA tool.

7.3 Future Research Directions

There is need for future work on both offensive and defensive side for malware encryption and propagation. We identify few research direction based on our research work.

- Security and protection of clear text loader(malware loader).
- Security analysis of Paillier-based malware encryption and re-randomization scheme.
- Encrypted malware auto propagation towards the target node.
- Paillier encryption key generation parameters p and q, formation from CSPRNG for malware encryption and auto-decryption.

References

[1] Philippe Beaucamps and Eric Filiol. On the possibility of practically obfuscating programs towards a unified perspective of code protection. *Journal in Computer Virology*, 3:3–21, 2006.

[2] Anthony Desnos. Implementation of k-ary viruses in python. *Hack. lu*, 2009.

[3] Thomas Dullien and Sebastian Porst. Reil: A platform-independent intermediate representation of disassembled code for static code analysis. 2009.

[4] Eric Filiol. Malicious cryptography techniques for unreversable (malicious or not) binaries. *CoRR*, abs/1009.4000, 2010. URL http://arxiv.org/abs/1009.4000.

[5] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Public Key Cryptography*, pages 112–121, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49162-0.

[6] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, pages 163–178, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24660-2.

[7] GreAT. Gauss: Abnormal distribution. Technical report, Kaspersky Lab Global Research and Analysis Team, Moscow, Russia, Aug 2012. URL https://securelist.com/gauss-abnormal-distribution/36620/.

[8] Eric Filiol. Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the bradley virus. Research Report RR-5250, INRIA, 2004. URL https://hal.inria.fr/inria-00070748.

[9] James Riordan and Bruce Schneier. *Environmental Key Generation Towards Clueless Agents*, pages 15–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-68671-2. doi: 10.1007/3-540-68671-1_2. URL https://doi.org/10.1007/3-540-68671-1_2.

[10] Herman Galteland and Kristian Gjøsteen. Malware encryption schemes - rerandomizable ciphertexts encrypted using environmental keys. *IACR Cryptology ePrint Archive*, 2017:1007, 2017.

[11] Darien Kindlund. Poison ivy: Assessing damage and extracting intelligence. Technical report, FireEye, Aug 2013. URL https://www.fireeye.com/blog/threat-research/2013/08/pivy-assessing-damage-and-extracting-intel.html.

[12] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003. ISBN 0131014056.

[13] Adware, spyware and other unwanted "malware" - and how to remove them. URL http://www.cexx.org/adware.htm.

[14] Cryptolocker ransomware infections: Cisa, Nov 2013. URL https://www.us-cert.gov/ncas/alerts/TA13-309A.

[15] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. ISSN 0001-0782. doi: 10.1145/359340.359342. URL http://doi.acm.org/10.1145/359340.359342.

[16] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[17] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA, 2009. AAI3382729.

[18] Xun Yi, Russell Paulet, and Elisa Bertino. *Homomorphic Encryption and Applications*. Springer Publishing Company, Incorporated, 2014. ISBN 3319122282, 9783319122281.

[19] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg, 1999. Springer-Verlag. ISBN 3-540-65889-0. URL http://dl.acm.org/citation.cfm?id=1756123.1756146.

[20] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[21] Burt Kaliski. *Euler's Totient Function*, pages 206–206. Springer US, Boston, MA, 2005. ISBN 978-0-387-23483-0. doi: 10.1007/0-387-23483-7_146. URL https://doi.org/10.1007/0-387-23483-7_146.

[22] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2005. ISBN 0131862391.

[23] Kristian Gjøsteen. Symmetric subgroup membership problems. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, pages 104–119, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-30580-4.

[24] Adam Young and Moti Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley; Sons, Inc., USA, 2004. ISBN 0764549758.

[25] Eric Filiol. *Computer viruses: from theory to applications*. 01 2005. ISBN 978-2-287-23939-7. doi: 10.1007/2-287-28099-5.

[26] Eric Filiol, Edouard Franc, Alessandro Gubbioli, Benoît Moquet, and Guillaume Roblot. Combinatorial optimisation of worm propagation on an unknown network. 2007.

[27] Eric Filiol and SÃľbastien Josse. A statistical model for undecidable viral detection. *Journal in Computer Virology*, 3:65–74, 05 2007. doi: 10.1007/s11416-007-0041-5.

[28] E Filiol. Malware of the future: when mathematics are on the bad sid. Hack.lu Conference, 10 2008. URL https://2018.hack.lu/archive/2008/Malware%20of%20the%20Future.pdf.

[29] Masahiro Mambo and Eiji Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. 1997.

[30] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69795-4.

[31] Lidong Zhou, Michael Marsh, Fred Schneider, and Anna Larsen-Redz. Distributed blinding for distributed elgamal re-encryption. *Proceedings - International Conference on Distributed Computing Systems*, 07 2004. doi: 10.1109/ICDCS.2005.24.

[32] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*, 2003.

[33] Zesheng Chen and Chuanyi ji. Spatial-temporal modeling of malware propagation in networks. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 16:1291–303, 10 2005. doi: 10.1109/TNN.2005.853425.

[34] Yini Wang, Sheng Wen, Yang Xiang, and Wanlei Zhou. Modeling the propagation of worms in networks: A survey. *IEEE Communications Surveys and Tutorials*, 16: 942–960, 2014.

[35] R. Rivest. The md5 message-digest algorithm. 1992.

[36] N.I.N.I.S. Technology. Sha-3 standard: Permutation-based hash and extendable-output functions: Fips pub 202. CreateSpace Independent Publishing Platform, 2015. ISBN 9781979406871. URL https://books.google.com.pk/books?id=hCwatAEACAAJ.

[37] Odzhan. odzhan/tinycrypt, Feb 2019. URL https://github.com/odzhan/tinycrypt/tree/master/stream/shake128.

[38] Ytisf. ytisf/thezoo, May 2017. URL https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/Ransomware.WannaCry.

[39] Ytisf. ytisf/thezoo, . URL https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/ZeusBankingVersion_26Nov2013.

[40] Mstfknn. mstfknn/malware-sample-library, Nov 2018. URL https://github.com/mstfknn/malware-sample-library/blob/master/GandCrab/.

[41] Ytisf. ytisf/thezoo, . URL https://github.com/ytisf/theZoo/blob/master/malwares/Binaries/Trojan.Kovter/.

[42] Ytisf. ytisf/thezoo, . URL https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/Linux.Wirenet.

[43] Ytisf. ytisf/thezoo, . URL https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/Linux.Encoder.1.

[44] Ashishb. ashishb/android-malware. URL https://github.com/ashishb/android-malware/tree/master/Dendroid.

[45] Jaydip Sen. *Homomorphic Encryption: Theory and Applications*. 07 2013. ISBN 978-953-51-1176-4. doi: 10.5772/56687.

[46] Shafrira Goldwasser. *Probabilistic Encryption: Theory and Applications (Partial Information, Factoring, Pseudo Random Bit Generation)*. PhD thesis, 1984. AAI8512835.

[47] L. Wang, K. Chen, Y. Long, and X. Mao. A new rsa-based proxy re-encryption scheme. *Journal of Computational Information Systems*, 11:567–575, 01 2015. doi: 10.12733/jcis13034.

[48] Ivan DamgÃěrd, Mads Jurik, and Jesper Nielsen. A generalization of paillierâĂŹs public-key system with applications to electronic voting. *International Journal of Information Security*, 9:371–385, 04 2003. doi: 10.1007/s10207-010-0119-9.

[49] 2019 sonicwall cyber threat report. URL https://www.sonicwall.com/lp/2019-cyber-threat-report-lp/.

[50] Robert Lyda and James Hamrock. Using entropy analysis to find encrypted and packed malware. *IEEE Security and Privacy*, 5(2):40–45, March 2007. ISSN 1540-7993. doi: 10.1109/MSP.2007.48. URL https://doi.org/10.1109/MSP.2007.48.

[51] Virustotal. URL http://www.VirusTotal.com/.

[52] Xabier Ugarte-Pedrero, Igor Santos, Borja Sanz, Carlos Laorden, and Pablo Bringas. Countering entropy measure attacks on packed software detection. 01 2012. doi: 10.1109/CCNC.2012.6181079.

[53] Peid cyptors and packet detector. URL http://www.aldeid.com/wiki/PEiD.

[54] Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of packed executables for accurate computer virus detection. *Pattern Recogn. Lett.*, 29(14):1941–1946, October 2008. ISSN 0167-8655. doi: 10.1016/j.patrec.2008.06.016. URL http://dx.doi.org/10.1016/j.patrec.2008.06.016.

[55] Igor Santos, Xabier Ugarte-Pedrero, Borja Sanz, Carlos Laorden, and Pablo G. Bringas. Collective classification for packed executable identification. In *Proceedings of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 23–30, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0788-8. doi: 10.1145/2030376.2030379. URL http://doi.acm.org/10.1145/2030376.2030379.

[56] Muhammad Zubair Shafiq, S. Momina Tabish, and Muddassar Farooq. Pe-probe: Leveraging packer detection and structural information to detect malicious portable executables. 2009.

[57] Scott Treadwell and Mian Zhou. A heuristic approach for detection of obfuscated malware. *2009 IEEE International Conference on Intelligence and Security Informatics*, pages 291–299, 2009.

[58] Ramandika Pranamulia, Yudistira Asnar, and Riza Perdana. Profile hidden markov model for malware classification - usage of system call sequence for malware classification. pages 1–5, 11 2017. doi: 10.1109/ICODSE.2017.8285885.

[59] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

[60] Robert Moskovitch, Yuval Elovici, and Lior Rokach. Detection of unknown computer worms based on behavioral classification of the host. *Computational Statistics and Data Analysis*, 52:4544–4566, 05 2008. doi: 10.1016/j.csda.2008.01.028.

[61] Mamoun Alazab, Sitalakshmi Venkatraman, Paul Watters, and Moutaz Alazab. Zero-day malware detection based on supervised learning algorithms of api call

signatures. In *Proceedings of the Ninth Australasian Data Mining Conference - Volume 121*, AusDM '11, pages 171–182, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc. ISBN 978-1-921770-02-9. URL http://dl.acm.org/citation.cfm?id=2483628.2483648.

[62] Muazzam Siddiqui, Morgan C. Wang, and Joohan Lee. A survey of data mining techniques for malware detection using file features. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, pages 509–510, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-105-7. doi: 10.1145/1593105.1593239. URL http://doi.acm.org/10.1145/1593105.1593239.

[63] Thomas H. Austin, Eric Filiol, Sébastien Josse, and Mark Stamp. Exploring hidden markov models for virus analysis: A semantic approach. *2013 46th Hawaii International Conference on System Sciences*, pages 5039–5048, 2013.

[64] Cisco encrypted traffic analytics. Jul 2019. URL https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.pdf.