



**UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA
EXPERIMENTAL YACHAY**

Escuela de Ciencias Matemáticas y Computacionales

**TÍTULO: Cryptography System Implementation Using
Neurocomputational Model and Deep Learning.**

Trabajo de integración curricular presentado como requisito para la
obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Valencia Ramos Rafael Alejandro

Tutor:

Ph.D. Chang Tortolero Oscar Guillermo

Urququí, diciembre del 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00039-AD

A los 27 días del mes de octubre de 2020, a las 14:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.
Miembro No Tutor	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

El(la) señor(ita) estudiante **VALENCIA RAMOS, RAFAEL ALEJANDRO**, con cédula de identidad No. **1724870843**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **CRYPTOGRAPHY SYSTEM IMPLEMENTATION USING NEUROCOMPUTATIONAL MODEL AND DEEP LEARNING.**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
--------------	--

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.	9,0
Presidente Tribunal De Defensa	Dr. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.	10,0
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0

Lo que da un promedio de: **9.7 (Nueve punto Siete)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

VALENCIA RAMOS, RAFAEL ALEJANDRO

Estudiante

DIEGO HERNAN PELUFFO ORDONEZ
Firmado digitalmente por DIEGO HERNAN PELUFFO ORDONEZ
 Fecha: 2020.11.23 12:19:37 -05'00'

Dr. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.

Presidente Tribunal de Defensa

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

Tutor



Firmado electrónicamente por:
OSCAR GUILLERMO CHANG TORTOLERO

CRISTHIAN RENE IZA PAREDES
Digitally signed by CRISTHIAN
RENE IZA PAREDES
Date: 2020.11.23 14:23:09
-05'00'
Dr. IZA PAREDES, CRISTHIAN RENE , Ph.D.
Miembro No Tutor

DAYSY MARGARITA MEDINA BRITO
Firmado digitalmente por DAYSY
MARGARITA MEDINA BRITO
Fecha: 2020.11.23 12:18:18 -05'00'
MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Rafael Alejandro Valencia Ramos**, con cédula de identidad 1724870843, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así como, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor(a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urququí, diciembre 2020.




Rafael Alejandro Valencia Ramos
CI: 1724870843

Autorización de publicación

Yo, **Rafael Alejandro Valencia Ramos**, con cédula de identidad 1724870843, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urququí, diciembre 2020.



Rafael Alejandro Valencia Ramos
CI: 1724870843

Dedication

To my family who has always supported me and encouraged me to improve myself and always keep moving forward.

Rafael Alejandro Valencia Ramos.

Acknowledgements

To Oscar Chang, an excellent teacher and an excellent person who has dedicated his time and effort in guiding the new generations.

To my family who has always supported me unconditionally in everything.

To all the teachers and friends who have guided and advised me, not only on an academic level but also on a human level. I couldn't have gotten this far without my family, friends and teachers.

Rafael Alejandro Valencia Ramos.

Resumen

El manejo y la protección de información siempre fueron temas de interés para la sociedad y de mucha importancia en la era digital actual. La gran mayoría de personas en todo el mundo ha manejado información importante y sensible a través de varias plataformas que van desde cuentas bancarias hasta redes sociales. Todas las plataformas han garantizado que la información que fluye por ellas se mantenga segura de ataques maliciosos. Esto ha dado lugar a un campo vasto e importante en la informática llamado criptografía. Existen varios algoritmos criptográficos que permiten que la información permanezca segura, y estos se han dividido en dos grupos, algoritmos de clave simétrica y clave asimétrica. El funcionamiento de los algoritmos de clave asimétrica está basado en la manipulación de números primos muy grandes, lo que proporciona un alto nivel de seguridad, pero también implica un elevado tiempo computacional. Este trabajo propone un sistema criptográfico basado en redes neuronales artificiales, implementado mediante el uso de técnicas de aprendizaje profundo. El método utiliza los pesos sinápticos de una red neuronal autocodificadora como claves de cifrado y descifrado, evitando el uso de números primos grandes. La solución propuesta permitió que los pesos sinápticos iniciales y finales, tengan un alto nivel de aleatoriedad, sin afectar el rendimiento general de la red. El análisis de seguridad teórico demostró que la metodología propuesta es robusta y difícil de romper. Los resultados experimentales confirmaron que el sistema propuesto realiza el cifrado y descifrado de datos en un bajo tiempo computacional, con respecto a algoritmos tradicionales como RSA, ElGamal, ECC y Paillier.

Palabras clave: autoencoder, criptografía, criptosistema, llaves de cifrado y descifrado, redes neuronales artificiales.

Abstract

The management and protection of information were always topics of interest for society and of great importance in the current digital age. The vast majority of people around the world have handled important and sensitive information through various platforms ranging from bank accounts to social media. All platforms have ensured that the information flowing through them is kept safe from malicious attacks. This has given rise to a vast and important field in computing called cryptography. There are several cryptographic algorithms that allow information to remain secure, and these have been divided into two groups, symmetric key and asymmetric key algorithms. The operation of asymmetric key algorithms is based on the manipulation of very large prime numbers, which provides a high level of security, but also implies a high computational time. This work proposes a cryptographic system based on artificial neural networks, implemented through the use of deep learning techniques. The method used the synaptic weights of an autoencoder neural network as encryption and decryption keys, avoiding the use of large prime numbers. The proposed solution allowed the initial and final synaptic weights to have a high level of randomness, without affecting the overall performance of the network. The theoretical security analysis indicated that the proposed methodology was robust and difficult to break. The experimental results confirmed that the proposed system performs the encryption and decryption of data in a low computational time, with respect to traditional algorithms such as RSA, ElGamal, ECC and Paillier.

Keywords: autoencoder, cryptography, cryptosystem, encryption and decryption keys, artificial neural networks.

Contents

List of Figures	12
List of Tables	14
1 Introduction	15
1.1 Problem Statement	16
1.2 Justification	16
2 Objectives	17
2.1 General Objective	17
2.2 Specific Objectives	17
2.3 Manuscript Outline	17
3 Technical Background	19
3.1 Cryptography	19
3.1.1 Basic Terms in Cryptography	19
3.1.2 Security Objectives	20
3.1.3 Classification of Cryptography	20
3.1.4 Asymmetric Key Algorithms	22
3.2 Artificial Neural Networks	25
3.2.1 Biological Foundations	25
3.2.2 From Natural Neural Neuron to Artificial Neuron	25
3.2.3 Activation Functions	26
3.2.4 Basic Network Structures	27
3.2.5 Autoencoder	28
3.2.6 Learning Process	29
3.2.7 Backpropagation Algorithm	30
4 Related Work	34
4.1 Cryptography Based on Neural Networks	34
4.1.1 First Approach	34
4.1.2 Cryptographic Properties of Neural Networks.	35
5 Methodology	37
5.1 Research Approach	37
5.2 Autoencoder Neural Network Architecture.	37
5.3 Randomization Algorithm	39

5.4	Asymmetric Key Cryptography Model	41
5.5	Encryption and Decryption Process	42
5.5.1	Encryption	42
5.5.2	Decryption	44
6	Performance and Security Parameters	46
6.1	Hardware Characteristics	46
6.2	Performance Parameters	46
6.2.1	Encryption Time	46
6.2.2	Decryption Time	46
6.2.3	Key Generation Time	47
6.3	Security Parameters	47
6.3.1	Precision	47
6.3.2	Network Tolerance	47
6.3.3	Private Key Security	47
6.3.4	Significant Digits Determination	47
6.3.5	Neural Network Security	48
6.3.6	Randomization Algorithm Efficiency	48
7	Results	49
7.1	Performance Evaluation	49
7.1.1	Encryption and Decryption Time of Cryptosystem	49
7.1.2	Key Generation Time of the Cryptosystem	50
7.1.3	Comparison between RSA, ElGamal and the proposed cryptosystem	51
7.1.4	Comparison between RSA-16bits, ECC-16bits and the proposed cryptosystem	55
7.1.5	Comparison between RSA, ElGamal, Paillier and the proposed cryp- tosystem	58
7.2	Security Evaluation	60
7.2.1	Private Key Security Results	60
7.2.2	Neural Network Security	61
7.2.3	Randomization Algorithm Performance	61
8	Conclusions	66
8.1	Future Work	66
	References	66

List of Figures

3.1	Secret or private key encryption scheme, which consists of a single key to carry out the encryption and decryption processes.	21
3.2	public-key encryption scheme, which consists of a different key for the encryption and decryption processes.	21
3.3	Scheme of a biological neuron which is composed of the dendrites, the axon and the pre-synaptic regions.	25
3.4	Diagram of the structure of a perceptron. The diagram shows all the parts that make up a perceptron and how the input information flows through each component until the output is generated.	26
3.5	General structure of an artificial neural network (ANN). Each circle represents a neuron and all connections represent the network weights. . . .	28
3.6	General structure of an autoencoder. Using the encoder f (mapping \mathbf{x} to \mathbf{h}) and the decoder g (mapping \mathbf{h} to \mathbf{r}), the autoencoder mapping an input \mathbf{x} to an output \mathbf{r} through an internal representation \mathbf{h}	28
3.7	Example of an autoencoder neural network architecture. The architecture is characterized by having the same number of input and output neurons. The number of neurons in the hidden layer is generally less than that of the input layer, however, this number of neurons may be higher than in the input layer.	29
5.1	Autoencoder architecture used in the proposed system. This architecture is composed of 8 neurons in both the input and output layers (following the definition of autoencoder) and the hidden layer contains 10 neurons. .	39
5.2	A representation of the randomization algorithm that shows the entire process that the initialization password undergoes. The algorithm is used to obtain several seeds with which the weight matrix of the neural network is initialized.	41
5.3	Scheme of the proposed cryptographic system which is composed of the initialization of the system, the creation of the keys and the data encryption/decryption process.	42
5.4	XOR text generation corresponding to the information preprocessing that the system performs.	43
5.5	Flow chart that shows how the ciphertext is obtained from the plain text. The diagram shows the information preprocessing and the use of the public key in the encryption process.	44

5.6	Flow chart showing how to get plain text from ciphertext. The diagram includes the inverse of preprocessing and how the private key is used in the decryption process.	45
7.1	Execution time corresponding to the encryption and decryption processes of the cryptographic system.	50
7.2	Time required by the cryptographic system to generate a set of keys with respect to the length of the initialization password.	51
7.3	Encryption times of the RSA, ElGamal algorithms and the proposed cryptosystem. The encryption times of RSA and ElGamal were obtained by Maqsood et al.	52
7.4	Decryption times of the RSA, ElGamal algorithms and the proposed cryptosystem. The decryption times for RSA and ElGamal were obtained by Maqsood et al.	53
7.5	Comparison of key generation times between RSA, ElGamal and the proposed Cryptosystem. The times of RSA and ElGamal were obtained by Maqsood et al.	54
7.6	Size comparison of the keys generated by RSA, ElGamal and the proposed Cryptosystem. RSA and ElGamal key sizes were obtained by Maqsood et al.	54
7.7	Encryption times of RSA-16bits, ECC-16bist and the proposed cryptosystem. The encryption times of RSA and ECC were obtained by Matta et al.	56
7.8	Decryption times of RSA-16bits, ECC-16bist and the proposed cryptosystem. The decryption times of RSA and ECC were obtained by Matta et al.	57
7.9	Comparison of key generation times between RSA-16bits, ECC-16bits and the proposed Cryptosystem. The times of RSA and ECC were obtained by Matta et al.	58
7.10	Encryption times of RSA, ElGamal, Paillier and the proposed cryptosystem. The encryption times of RSA, ElGamal and Paillier were obtained by Farah et al.	59
7.11	Decryption times of RSA, ElGamal, Paillier and the proposed cryptosystem. The decryption times of RSA, ElGamal and Paillier were obtained by Farah et al.	60
7.12	Pseudo-order index vector distribution with passwords “helloworld” and “helloworle”	62
7.13	Public and Private key distribution with password “helloworld”	63
7.14	Public and Private key distribution with password “helloworle”	63
7.15	Comparison between public keys generated by passwords “helloworld” and “helloworle”	64
7.16	Comparison between private keys generated by passwords “helloworld” and “helloworle”	65

List of Tables

3.1	Paillier's Scheme 1	24
3.2	Paillier's Scheme 3	24
3.3	Activation Functions	27
5.1	Example of message encryption	43
5.2	Example of message decryption	45
7.1	Key Generation Time and Keys Size of the Cryptosystem	51
7.2	Encryption Time comparative with Maqsood et al.	52
7.3	Decryption Time comparative with Maqsood et al.	53
7.4	Key Generation Time comparative with Maqsood et al.	54
7.5	Encryption Time comparative with Matta et al.	55
7.6	Decryption Time comparative with Matta et al.	56
7.7	Key Generation Time comparative with Matta et al.	57
7.8	Encryption Time comparative with Farah et al.	58
7.9	Decryption Time comparative with Farah et al.	59
7.10	Network Tolerance	60
7.11	Neural Network Security Results	61

Chapter 1

Introduction

Information security refers to the process of protecting digital information, preventing access, use, disclosure, modification or destruction of data without authorization [1]. Nowadays, no system can guarantee 100% data protection and information security has become a topic of great interest in the academic and industrial field. Since the massive increase of people with internet access, the information vulnerable to attacks has also become massive and data transport systems have been forced to improve their data protection. The existing or available systems are based on the concept of cryptography, which is considered both science and an art [2].

Cryptography-based systems play a critical role in the process of ensuring a secure communication between multiple entities, based on data encryption and decryption. The term encryption refers to transforming a simple message into an equivalent that cannot be read or understood, known as encrypted text. In addition, when we talk about decryption, we are referring to the process that transforms the encrypted text back to the original text [3].

Cryptography can be classified into symmetric key cryptography and asymmetric key or public-key cryptography [4]. Symmetric key encryption consists of using the same key for encryption and decryption process. The algorithms that are used to symmetric encryption are generally based on data substitution and permutation [5]. Symmetric key algorithms include DES (Data Encryption Standard), 3DES (Triple Data Encryption Standard), AES (Advanced Encryption Standard), Blowfish Encryption Algorithm, International Data Encryption Algorithm, etc [4]. On the contrary, asymmetric key encryption uses two keys in the process of the data codification. The public-key cryptographic algorithms are based on mathematical equations and use one key for encryption and a different key for decryption. Public-key algorithms are RSA (Rivest, Shamir and Adleman), Elgamal and ECC (elliptic curve cryptography) [5].

In addition, artificial intelligence and neural networks provide a tool for the development of complex works on the information security field, such as detection of credit card fraud [6], image cryptography [7], communication protection with adverse neural cryptography [8], etc.

In this work by using an ANN called Autoencoder, two keys are generated, one public and one private. Through random training which requires an initialization password, the system is trained using the full ASCII code, which consists of 256 characters. To measure the effectiveness of the proposed system, it is compared with other public-key

algorithms, using encryption, decryption and key generation times as metrics. Besides, a security analysis is carried out to determine the difficulty that exists to break the proposed security of the system.

1.1 Problem Statement

The problem statement lies in that most asymmetric data encryption algorithms are serial and present a considerable computational cost since they use complex formulas and large prime numbers. The objective of this research thesis is the development of an asymmetric key cryptographic system capable of encrypting messages in lower computation time with respect to traditional algorithms, and guaranteeing high security for encrypted data. The formulation of the problem can be divided into three parts: (1) Encode the 256 characters of the complete ASCII code using an autoencoder neural network which integrates a randomization algorithm that allows increasing the randomness in the training of the network and therefore, in the generation of the public and private keys. (2) Evaluate and establish the level of security presented by the proposed system, in addition. (3) Measure the performance of the system by comparing it with other asymmetric key algorithms.

1.2 Justification

Data security has a crucial role in today's digital world, for this reason, the computational performance and security of encryption algorithms must be taken into account. Generally, traditional asymmetric key algorithms have a serial behavior and use complex mathematical operations such as long prime-number calculations and discrete mathematics. These operations require high computational power. For these reasons, this thesis intends to implement artificial intelligence techniques (artificial neural networks) that allow generating a system that presents a better computational performance and high security with respect to traditional algorithms. In this case, neural networks have an advantage since they can increase randomness (security), and simplify the operations used in data encryption and decryption.

Chapter 2

Objectives

2.1 General Objective

To develop an asymmetric cryptography system based artificial intelligence and using an artificial neural network, capable of being a viable alternative to encode and decode information using asymmetric keys.

2.2 Specific Objectives

- To find an efficient autoencoder neural network architecture, capable of encoding the complete **ASCII** code in reasonable computation time.
- To design a process that allows the training of the neural network (key generation) with a high randomized level.
- To implement a strategy to guarantee that the information encoded by the system is difficult to violate, and prove that the system has a high level of security.

2.3 Manuscript Outline

This work is structured into the following chapters:

- **Chapter 1 - Introduction.** The introduction, the problem statement and the justifications of this work are presented.
- **Chapter 2 - Objectives.** The general objective and the specific objectives of this work are established.
- **Chapter 3 - Technical Background.** A technical background about artificial intelligence and information security approaches are presented.
- **Chapter 4 - Related Work** presents a review of recent work related to information security developed with artificial neural networks.

- **Chapter 5 - Methodology.** The cryptographic system proposal is presented in detail.
- **Chapter 6 - Performance and security parameters.** The performance of our proposal and the security parameters in its evaluation are described.
- **Chapter 7 - Results** present the main findings of proposed system.
- **Chapter 8 - Conclusions**

Chapter 3

Technical Background

This chapter presents the concepts that established the bases for the development of an asymmetric key cryptographic system capable of combining the cryptography and the artificial intelligence.

3.1 Cryptography

Cryptography refers to the study of information hiding and retrieval [9]. Data hiding is known as “data encryption” and retrieval is defined as “decryption” [10]. Cryptography comes from the Greek words *kryptós* that means “hidden” and *gráphein* that refers to “to write”, therefore, cryptography means “hidden writing”. The process of hiding data is considered an art, which aims to protect the information through a transformation to an unreadable format for all readers except the receiver, who is capable to recover the original information from the unreadable format. The main objective of cryptography is to keep information safe from theft or unauthorized access [9].

3.1.1 Basic Terms in Cryptography

Cryptography has several terms that are useful to understand this area of study and among these terms are:

1. Plain text: The original message or data that will be entered into the encryption algorithm [11].
2. Encryption: The process of converting plain text into encrypted text. This process is done on the sender’s side and requires two things, an encryption algorithm and a secret key [9, 10].
3. Encryption algorithm: It is responsible to perform substitutions and transformations in plain text [11].
4. Secret key: A value external to the encryption algorithm and different from plain text. This value can be a numeric, alphanumeric text or a special symbol [10]. The transformation and substitutions that the algorithm performs will depend on the key, therefore, a different result will be produced depending on the key [11].

5. Ciphertext: The encrypted message generated with the encryption algorithm and the key. A plain text will have two different “representations” if two different keys are used. The ciphertext is an unintelligible “random” data set [11].
6. Decryption: The process of converting the ciphertext into plain text. This process is done on the receiver side to retrieve the original message from the ciphertext [10]. To perform the decryption, a decryption algorithm and a key are required [9].
7. Decryption algorithm: It can be seen as the encryption algorithm executed in reverse, which transforms the ciphertext into plain text using the secret key [10].
8. Symmetric encryption or private key: It is an encryption method that uses the same key to encrypt and decrypt the data. This method requires that the sender and the receiver have the same key which must remain private [12].
9. Asymmetric encryption or public-key cryptography: It is an encryption method that uses two different keys, one for the encryption process and another for the decryption process. These two keys are called the public key and the private key. This method requires that the private key remain secret, while the public key does not present any risk and is of public domain [12].
10. Encoder: It is the person who sends the message and uses an encryption algorithm to make the message secure [9].
11. Decoder: It is the person capable of using a decrypting algorithm to decipher the message. This person can be the recipient of the message or an intruder who tries to read the encrypted message [9].

3.1.2 Security Objectives

Cryptography has many security objectives such as confidentiality, authentication, integrity, non-repudiation, and access control. Confidentiality refers that the transmitted information can only be read by the receiver. Authentication is the identification process that determines the provenance of the information, whether the information comes from an authorized person or an unauthorised person. Integrity means that no one other than the sender and receiver can alter the information [13]. Non repudiation, guarantees that the information transmission cannot be denied by either the sender or the receiver. Access control, guarantees that only authorized persons can access to the information [10].

3.1.3 Classification of Cryptography

Cryptography can be divided into three types, secret key cryptography (SKC), public-key cryptography (PKC), and hash functions [14].

Secret Key Cryptography

This type of cryptography is also known as symmetric cryptography, due to it uses a single key to encrypt and decrypt the data (See Figure 3.1). The most popular secret key algorithms are Data Encryption Standard (DES), triple-DES, Rivest Cipher 2 (RC2), Blowfish, CAST, Serpent, TEA, AES (Rijndael), Twofish, IDEA, RC-6 and MARS [15]. This cryptography requires that the key be known by both the sender and the receiver, for this reason, the difficulty of this approach lies in the distribution of the secret key. Secret key encryption is faster compared to public-key, however, key distribution can become a complicated logistical problem [13].

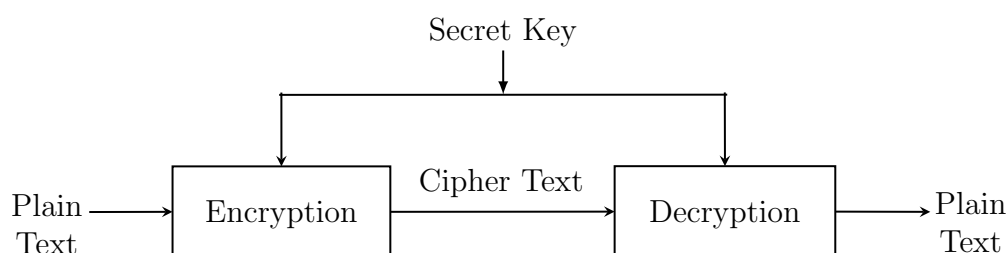


Figure 3.1: Secret or private key encryption scheme, which consists of a single key to carry out the encryption and decryption processes.

Public-key Cryptography

This type of cryptography is also known as asymmetric key cryptography and is based on mathematical functions computationally intensive [15] (See Figure 3.2). This is an encryption scheme that uses two mathematically related, but not identical keys known as a public key and a private key. Unlike symmetric key encryption approaches that rely on a unique key to encrypt and decrypt, in the public-key cryptography each key performs a unique function [14]. The public key is used in the encryption process and the private key is used in the decryption process [13]. There are various popular asymmetric cryptography algorithms such as RSA, Diffie-Hellman keys, ElGamal, etc [15].

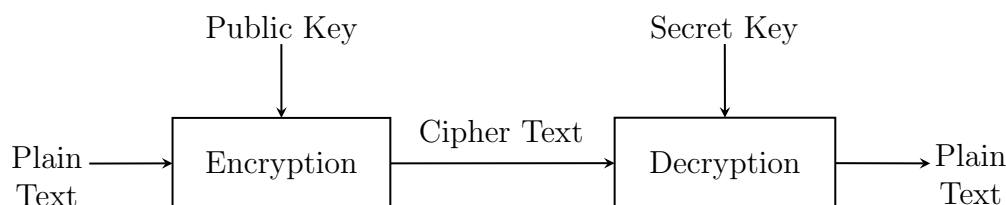


Figure 3.2: public-key encryption scheme, which consists of a different key for the encryption and decryption processes.

Hash Function

Hash function or one-way encryption: performs a non-reversible mathematical transformation that assigns a unique digest or finger print to the given data [15]. Hash functions do not use a key since they calculate a fixed-length hash value that depends on the plain text. This process makes it impossible to retrieve the content of the original message. The hash function has three properties:

- It is very easy calculate a hash for any data.
- In computational terms, it is very difficult to calculate an alphanumeric text that has a certain hash.
- It is unlikely that two different messages will have the same hash.

Hash functions are used primarily by operating systems to perform password encryption. Additionally, these functions are used in various applications such as message integrity checks, digital signatures, authentication, among other security applications. The most commonly used hash functions are MD5 and SHA-1 [14].

3.1.4 Asymmetric Key Algorithms

Rivest-Shamir-Adleman (RSA)

RSA was developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA is an asymmetric or public-key cryptosystem based on prime number theory. The security of the system is based on the difficulty that exists in the decomposition of large prime numbers, which are considered mathematical problems that do not have an effective solution [16, 17]. The RSA cryptosystem is one of the most typical method used for public-key cryptography and is considered the first algorithm used for both data encryption and the creation of digital signatures [17]. The RSA algorithm consists of three steps encryption, decryption and key generation. RSA has to deal with different kind of attacks such as mathematical attacks, brute-force attacks, chosen ciphertext attacks and timing attacks [16].

The RSA cryptosystem uses a three different positive numbers n , e and d , such that:

$$(m^e)^d \equiv m(\text{mod } n)$$

Encryption

$$C = E(M) \equiv M^e(\text{mod } n)$$

Decryption

$$M = D(C) \equiv C^d(\text{mod } n)$$

where,

M = Message or Plain text

C = Ciphertext

(n, e) = Public Key

(n, d) = Private Key

Diffie-Hellman

The Diffie-Hellman algorithm is a public-key algorithm developed by Witfield Diffie and Martin Hellman in 1976. The algorithm uses a key exchange algorithm and was developed under an insecure communication channel [16]. This algorithm consists of two keys, one private and one public, that are used in the encryption and decryption process of the information (public key in encryption and private key in decryption) [18].

ElGamal

The ElGamal scheme was proposed by T. ElGamal in 1984 [19]. ElGamal is an asymmetric encryption scheme based on the key exchange process established in the Diffie-Hellman algorithm. This encryption scheme is made up of three algorithms used for key generation, encryption and decryption. This process is described below [20]:

Key Generation:

1. Let p a large prime number, and a a primitive element ($\text{mod } p$).
2. Select a random number x , and compute $y \equiv ax(\text{mod } p)$.
3. Publish the public key (a, p, y) , and keep x as the secret key.

Encryption:

1. Select a random number r such that $r \in Z_q$, and then compute $C_1 \equiv a^r(\text{mod } p)$.
2. Compute $C_2 \equiv m \times y^r(\text{mod } p)$.
3. Then consider (C_1, C_2) as the ciphertext.

Decryption:

1. Compute $m \equiv C_2 \times C_1^{-x}(\text{mod } p)$.

Elliptic Curve Cryptography (ECC)

ECC was developed in 1985 by Koblitz and Miller [16]. ECC is a type of public-key cryptography, based on the algebraic structure of elliptic curves in finite fields [21]. ECC uses a complex algebraic and geometric equations for key generation. The ECC scheme uses the private key for both the decryption process and the generation of signatures. The public key generated by ECC is used for both encryption and signature verification [16]. ECC can be used as a complement to other encryption algorithms thus generating new ones such as ECC-Diffie-Hellman and ECC-DSA [22].

An elliptical curve is the set of points described by the equation:

$$E_p(a, b) : y^2 = x^3(\text{mod } p),$$

where p is a prime number.

Paillier

Paillier was described in 1999 under the name *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes* by Pascal Paillier. This cryptosystem has two versions of a partially homomorphic cryptographic system. The basic version is called Scheme 1 (See Table 3.1), and the faster decryption variant is called Scheme 3 (See Table 3.2) [23]. The security of the Paillier scheme depends on the integer factorization hardness and is based on the n -th residues in $\mathbb{Z}_{n^2}^*$. Paillier's schemes can be described in five parts parameters, public key, private key, encryption and decryption [24].

Table 3.1: Paillier's Scheme 1

Parameters	prime numbers p, q $n = pq$ $\lambda = lcm(p - 1, q - 1)$ g , with $g \in \mathbb{Z}_{n^2}^*$ (the order of g multiple of n)
Public Key	n, g
Private key	p, q, λ
Encryption	plain text $m < n$ choose random number $r < n$ such that $r \in \mathbb{Z}_n^*$ ciphertext $c = g^{m r^n} \pmod{n^2}$
Description	ciphertext $c < n^2$ plain text $m = \frac{L(c^\lambda \pmod{n^2})}{L(g^\lambda \pmod{n^2})} \pmod{n}$

Table 3.2: Paillier's Scheme 3

Parameters	prime numbers p, q $n = pq$ $\lambda = lcm(p - 1, q - 1)$ α , a divisor of λ $g \in \mathbb{Z}_{n^2}^*$ (the order of g multiple of αn)
Public Key	n, g
Private key	p, q, α
Encryption	plain text $m < n$ choose random number $r < \alpha$ ciphertext $c = g^m (g^n)^r \pmod{n^2}$
Description	ciphertext $c < n^2$ plain text $m = \frac{L(c^\alpha \pmod{n^2})}{L(g^\alpha \pmod{n^2})} \pmod{n}$

3.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational networks that try to simulate the decision process of the biological neuron networks that conform the central nervous system of a living being [25]. Among the first studies on artificial neural networks, we have Warren McCulloch and Walter Pitts, who in 1943 published an article where purpose use ANNs as a way to simulate the human brain. Later, Minsky and Dean Edmonds, in 1951 developed the stochastic neural analog reinforcement calculator, which is recognized as the ANN [26].

3.2.1 Biological Foundations

The biological neural network consists of thousands of neurons (nerve cells) interconnected with each other. In Figure 3.3 the graphic representation of a biological neuron is shown. The nucleus of the cell is the place where most of the “calculations” are performed. Once the information is processed by the nucleus, the information is distributed through synaptic connections among all interconnected neurons [27].

The information distribution process takes place in the pre-synaptic region, which is made up of several membranes connected to the axon. These membranes communicate with the dendrites of other neurons through electrochemical signals, which can excite the cell or inhibit its activation [25, 28].

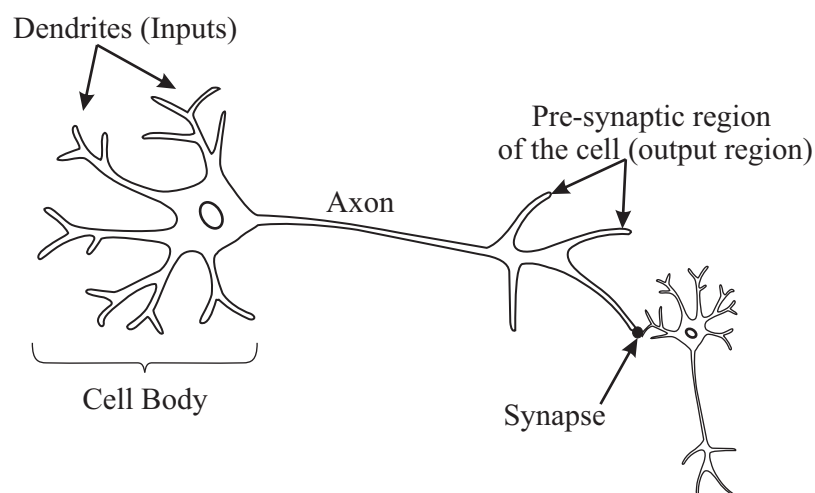


Figure 3.3: Scheme of a biological neuron which is composed of the dendrites, the axon and the pre-synaptic regions.

3.2.2 From Natural Neural Neuron to Artificial Neuron

In an analog sense, artificial neural networks follow the same information flow as natural neural networks. The mathematical model that describes the artificial neural networks is very simple in comparison to the complex behavior of the biological neural networks. The ANNs are conformed by nodes that fulfill the role of neurons. These nodes are

interconnected and distribute information through functional links [28]. The simplest model of a neuron is called a “perceptron” and its representation is shown in Figure 3.4.

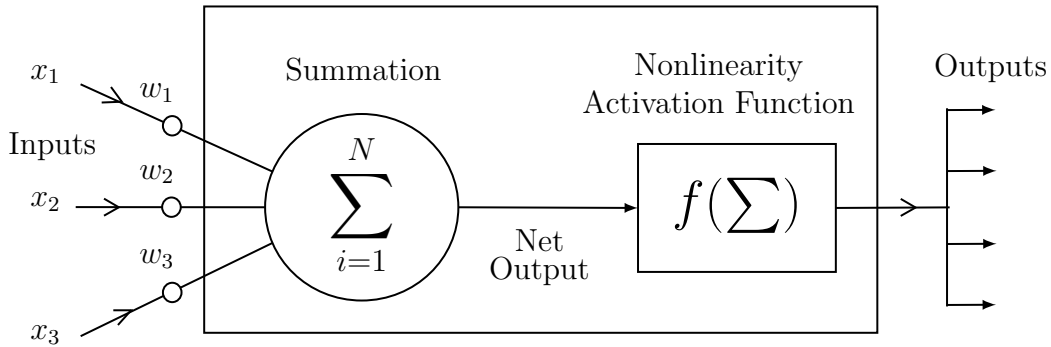


Figure 3.4: Diagram of the structure of a perceptron. The diagram shows all the parts that make up a perceptron and how the input information flows through each component until the output is generated.

The artificial neuron consists of an input \mathbf{X} , and a single output \mathbf{o} .

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n].$$

Each input \mathbf{x}_i is multiplied by a weight function w_i and the resulting value is added from $i \in \{1, \dots, N\}$ (See Equation 3.1). This process is called the net input of the neuron, which can be expressed as follows:

$$net = \sum_{i=1}^N w_i \mathbf{x}_i. \quad (3.1)$$

Then the activation function is computed. The most common nonlinear activation function is the sigmoid, however, there are several activation functions such as hard-limiter, arc-tangent, hyperbolic tangent sigmoid, etc [28]).

3.2.3 Activation Functions

Among the most popular activation functions are Binary Step Function, Linear, Sigmoid, Tanh, ReLU, Swish, SoftMax, etc. These functions play a fundamental role in the process of transformation of the net output of the neuron, which activate or inhibit the neurons with which they are connected [29]. Some activation functions are shown in the Table 3.3.

Table 3.3: Activation Functions

Binary Step Function	$f(x) \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$
Linear	$f(x)=ax$
Sigmoid	$f(x)=1/1+e^{-x}$
ReLU	$f(x) \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$
Tanh	$f(x)=2/1+e^{-2x} - 1$
Swish	$f(x) = x \times \text{sigmoid}(x)$ $f(x) = x/(1 + e^{-x})$

3.2.4 Basic Network Structures

The complete structure of an ANN is composed of at least three layers the input layer, the hidden layer and the output layer. Each layer is conformed by a certain number of neurons that are interconnected with all neurons that conform other layers (See Figure 3.5). Each connection represents a weight which is modified according to the activation function used to the network setting. Once the weights reach an optimal value, the network is said to be trained [30].

Historically, the perceptron proposed by McCulloch and Pitts is the first artificial neuron model and is considered the most abstract form of a neural network [31, 27]. Later, ADALINE (adaptive linear network) is developed by Widrow and Hoff. ADALINE continues to be a single neuron, which is adaptively trained to reduce error through an adjusting of the weights in the neural network [32]. Next Madanile (Many Adeline) is developed, which is a multilayer neural network formed by several adalines. These early ANN models, especially the Perceptron, are the basis for almost all ANN architectures [27].

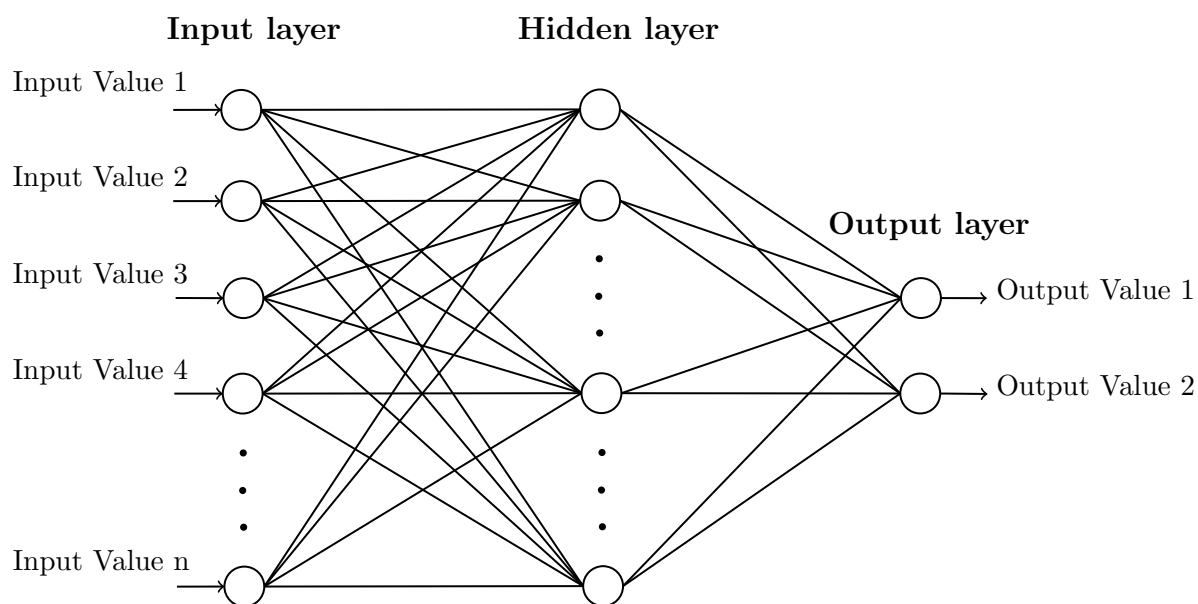


Figure 3.5: General structure of an artificial neural network (ANN). Each circle represents a neuron and all connections represent the network weights.

3.2.5 Autoencoder

An autoencoder (AE) is a symmetric neuronal network capable of learning to reconstruct the input X from characteristics [33].

The general structure of an autoencoder consists of two parts, the encoding function $h = f(x)$ and the decoding function $g(f(x)) \approx x$ [34].

The autoencoder is designed to work with approximations and to learn to copy perfectly $g(f(x)) = x$ for all points. The general structure of the autoencoder is shown in Figure 3.6.

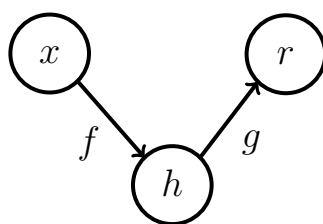


Figure 3.6: General structure of an autoencoder. Using the encoder f (mapping \mathbf{x} to \mathbf{h}) and the decoder g (mapping \mathbf{h} to \mathbf{r}), the autoencoder mapping an input \mathbf{x} to an output \mathbf{r} through an internal representation \mathbf{h} .

Autoencoders, due to their architecture, are restricted to approximate the input values of the network by prioritizing only the useful properties extracted from the input data [35]. In other words, autoencoders perform an input data re-dimensionalization, depending on the number of hidden neurons of the neural network. Once the data entered into the network is re-dimensionalized by the encoding function, the generated “code” is mapped

through the decoding function. Autoencoders use an unsupervised training since is not required labeled data in the training process [36]. Figure 3.7 shows an example of the autoencoder artificial neural network architecture.

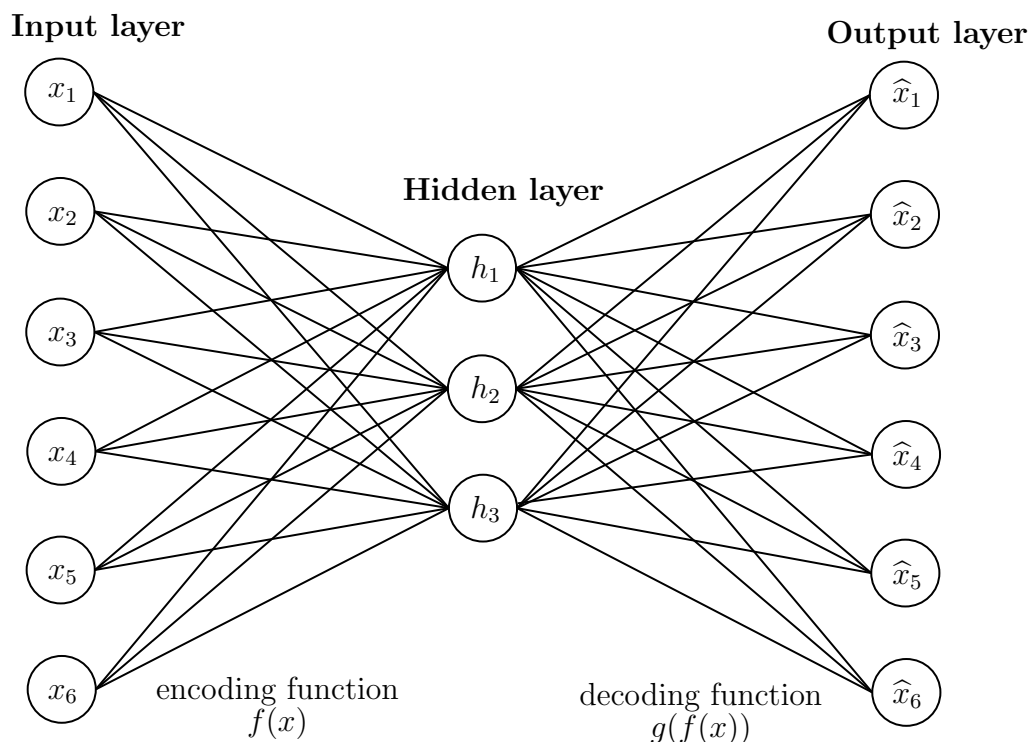


Figure 3.7: Example of an autoencoder neural network architecture. The architecture is characterized by having the same number of input and output neurons. The number of neurons in the hidden layer is generally less than that of the input layer, however, this number of neurons may be higher than in the input layer.

3.2.6 Learning Process

So that neural networks can be capable to solve a certain task, instead of a directly programmed they are trained by a learning algorithm. The learning algorithms are important since they specify the final configuration of the neural network, in other words, the algorithms condition and determine the capacity of the network [37].

There are several learning algorithms, some require a monitoring method that verifies the optimal output of the network, others are unsupervised algorithms that do not need a verification of the output of the network. The learning process can be classified into supervised, unsupervised, and reinforcement [38, 37].

The most common learning rules are Hebbian learning rule, Correlation learning rule, Instar learning rule, Winner Takes All (WTA), Outside Learning rule, Error Backpropagation learning, etc [38].

- Unsupervised: In this type of learning, the network is trained based on a set of inputs, without providing information to guide the outputs of the network.

- Supervised: In this type of training, need to identify the set of examples. A set of input and output examples are presented to the network, to it can learn from them.
- Reinforcement: This training is used when the inputs and outputs of the network cannot be specified.

The data set used for learning a neural network is defined as a learning or training set [37].

3.2.7 Backpropagation Algorithm

The Backpropagation algorithm (BP algorithm) is used to perform supervised training [37]. The algorithm trains the weights of a multilayer network using gradient descent to minimize the squared error between the network inputs and the target parameters (network outputs) [39]. The algorithm assigns a small initial value to the network connection, and the error gradient relative to the sample is calculated from a training sample of the network [27].

The learning process of the backpropagation algorithm can be divided in two parts, forward propagation and backpropagation [40, 39, 27].

1. Forward propagation of information: The input information of the network propagates from the input layer to the output layer through all hidden layers. During this propagation the weights operating signals and offset value of the network remain constant and the state of each neuron layer only affects the next layer. If the output network does not is the expected one, an inverse propagation is performed with the error information. The error function of the network output units is defined by:

$$E = \frac{1}{2} \sum_{d \in D} \sum_{k \in N} (t_{kd} - O_{kd})^2, \quad (3.2)$$

where N is the set of output neurons in the network, t_{kd} and O_{kd} are the target and output values of the k -th output unit in the training example d .

2. Backpropagation of error information: In this process, the error information propagates from the output layer to the input layer through all the hidden layers. During this propagation, the values of the weights of the network are modified with respect to the calculated error. This process makes it possible to increase the similarity of the network output value with the expected value.

Mathematical Description of the BP Algorithm

The mathematical formula [40] of the BP model is presented as follows:

Forward propagation: Output of the network.

- Hidden layer node output

$$y_j = f\left(\sum_i w_{ji}x_i\right) = f(\text{net}_j) \quad (3.3)$$

- Output layer node output

$$z_l = f\left(\sum_j v_{lj}y_j\right) = f(\text{net}_l) \quad (3.4)$$

- Output node error

$$E = \frac{1}{2} \sum_l (t_l - z_l)^2 \quad (3.5)$$

Back propagation: Modification of weight value

- Output node derivation by means of error function

$$\frac{\partial E}{\partial v_{lj}} = \sum_{k=1}^n \frac{\partial E}{\partial z_k} \times \frac{\partial z_k}{\partial v_{lj}} = \frac{\partial E}{\partial z_l} \times \frac{\partial z_l}{\partial v_{lj}}, \quad (3.6)$$

where

$$\begin{aligned} \frac{\partial E}{\partial z_l} &= \frac{1}{2} \sum_k [-2(t_k - z_k) \times \frac{\partial z_k}{\partial z_l}] = -(t_l - z_l) \\ \frac{\partial z_l}{\partial v_{lj}} &= \frac{\partial z_l}{\partial \text{net}_l} \times \frac{\partial \text{net}_l}{\partial v_{lj}} = f'(\text{net}_l) \times y_j. \end{aligned}$$

So,

$$\frac{\partial E}{\partial v_{lj}} = -(t_l - z_l) \times f'(\text{net}_l) \times y_j.$$

Suppose the error of the input node is,

$$\delta_l = -(t_l - z_l) \times f'(\text{net}_l).$$

Therefore,

$$\frac{\partial E}{\partial v_{lj}} = -\delta_l \times y_j. \quad (3.7)$$

- Hidden layer node output deviation by error function

$$\frac{\partial E}{\partial w_{ji}} = \sum_l \sum_j \frac{\partial E}{\partial z_l} \times \frac{\partial z_l}{\partial y_j} \times \frac{\partial y_j}{\partial w_{ji}}, \quad (3.8)$$

where,

$$\frac{\partial E}{\partial z_l} = \frac{1}{2} \sum_k [-2(t_k - z_k) \times \frac{\partial z_k}{\partial z_l}] = -(t_l - z_l).$$

So,

$$\frac{\partial E}{\partial w_{ji}} = - \sum_l (t_l - z_l) \times f'(\text{net}_l) \times v_{lj} \times f'(\text{net}_j) \times x_i = - \sum_l \delta_l v_{lj} f'(\text{net}_j) \times x_i.$$

Suppose that the error of hidden layer node is:

$$\delta'_j = f'(net_j) \times \sum \delta_l v_{lj}.$$

Therefore,

$$\frac{\partial E}{\partial w_{ji}} = -\delta'_j x_i. \quad (3.9)$$

Due to, the modification of weight Δv_{lj} and Δw_{ji} is proportional to the error functions and descends along the gradient, the formula that describe the modification of weight of hide layer and output layer is:

$$\Delta v_{lj} = -\eta \frac{\partial E}{\partial v_{lj}} = \eta \delta_l y_j, \quad (3.10)$$

where η is the learning rate.

The formula that describes the modification between the input layer and the hidden layer is:

$$\Delta w_{ji} = -\eta' \frac{\partial E}{\partial w_{ji}} = \eta' \delta'_j x_i,$$

where,

$$\delta'_j = f'(net_j) \times \sum_l \delta_l v_{lj}.$$

Here η' represent the learning rate, δ'_j expresses the error δ_l of the output node z_l that is propagated through the v_{lj} weight value to the y_j node, becoming the node error of the hidden layer.

Pseudo Code of BP Algorithm.

The pseudo code [39] of a backpropagation-trained neural network is presented below.

```

1. Create a neural network with  $n_{in}$  inputs neurons,  $n_{out}$  output neurons, and
 $n_{hid}$  hidden neurons.
2. Initialize the weights with small values ( $w_{ij} \in [-0.5, 0.5]$ ) .
while not training do
  for each training example  $(x, t)$ , where  $x$  is the input and  $t$  is the target value.
  do
    if Propagate the input forward in the network: then
      for every neuron in the network do
        Input the  $x$  value to the network and calculate the output  $o_u$ 
      end
    end
    if Propagate the error backward in the network: then
      for every output neuron  $k$  do
        Calculate the error term  $\delta_k$ :
         $\delta_k \geq o_k(1 - o_k)(t_k - o_k)$ 
      end
      for each hidden neuron  $h$  do
        Calculate the error term  $\delta_h$ :
         $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$ 
      end
      for each neuron weight do
        Update  $w_{ji}$ :
         $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ 
        where
         $\Delta w_{ji} = \eta \delta_j x_{ji}$ 
      end
    end
  end
end

```

Algorithm 1: Backpropagation Algorithm.

Chapter 4

Related Work

This chapter presents the previous work and the approaches with which they combine cryptography with artificial intelligence.

4.1 Cryptography Based on Neural Networks

According to Charniya [41], neural networks are able to accurately identify a nonlinear system model from the inputs and outputs of a complex system and do not need to know the exact relationship between inputs and outputs. By extending these concepts to the field of cryptography, neural networks could produce encryption keys with high-level security [42]. In [43], a neural network is established as the only algorithm or method for key generation through a public channel that is not based on number theory. The main advantages over traditional approaches are simple, low calculations for training and a new key can be generated for each message exchange.

4.1.1 First Approach

In [44], it is determined that the use of autoencoder neural network as encoder and decoder of data is feasible. In this article, several comparisons between network architectures are made, to determine which ones are capable of storing an efficient binary encoding of the ASCII code. To determine a suitable architecture, two data sets, one with 52 characters and the other with 95 characters, were coded in network architectures in which the number of neurons in the hidden layer varied. The results showed that an autocoder with more neurons in the hidden layer than in the input/output layers had better accuracy.

In [45], it is developed in a symmetric encryption system where the neural network is used as a data encryption system, which by a single initialization key can encode and decode data. The architecture shows that the work is viable and a strong starting point.

Taking the concepts established in [44] and [45], this work proposes a new approach which seeks to develop an asymmetric key system capable of increasing security by separating the neural network and using each part as different encryption and decryption keys. The use of two different keys and a randomization algorithm capable of not altering the performance of the network, give the possibility of increasing the randomness in the training of the network and consequently the security of the system.

4.1.2 Cryptographic Properties of Neural Networks.

Volna [46] in his work on neural network-based cryptography, mention that ANNs offer a very powerful general framework for representing the non-linear mapping of various input variables to various output variables. Based on this concept, an encryption system using a permanently changing key was developed. In Volna's work, the topology is a very important issue to achieve a correct operation of the system, so a multilayer topology was implemented, considered the most suitable topology in this case. The encryption and decryption process is done using the backpropagation algorithm.

Jogdand [47] proposed that neural networks can be used to generate common secret keys. In this work on neural cryptography, two neural networks are defined, which receives an identical input vector, with which they are trained to generate an output bit. Both networks and their weight vectors exhibit a novel phenomenon, in which the networks are synchronized in a time-dependent state with identical weights. The secret key generated through a public channel is used to encrypt and decrypt the information that is sent through the channel.

Kinzel and Kanter [48] show the application of interactive neural networks for the exchange of keys through a public channel. This work shows that two neural networks trained simultaneously, achieve a state of synchronization in which the values of their synaptic weights are identical depending on the training time. This neural cryptography uses a topology called the tree parity machine that consists of one output neuron, K hidden neurons, and $K \times N$ input neurons. The hidden values are equal to the sign function of the dot product of the input and the weights, while the output value is the multiplication of the hidden values. It has not been proven that there is no algorithm for the successful attack, but this approach is very difficult to crack by brute force. Even if an attacker knows the input/output relationship along with the algorithm, it could not recover the secret common key that A and B use for encryption.

Klein et al. [49] shows the applications of mutual learning neural networks that synchronize their time-dependent weights. It suggests that synchronization is a novel approach to generating a secure cryptographic secret key using a public channel. This work describes the learning process in a simple network, where two perceptrons receive a common and change their weights according to their mutual output, and the learning process in a tree of parity machines. Klein, to understand more about synchronization, the process is analytically described using statistical physics methods, developing a new technique that combines neural networks with chaos synchronization. This allows having a more secure system against attackers, since the generation of pseudo-random numbers increases the security of a cryptosystem. [50].

A field in which neural networks have been applied together with cryptography is the steganography. This field can be defined as a technique to hide messages within other messages [51]. Additionally, the steganalysis is the art and science of detecting whether a given medium has a hidden message in it [52].

In [53] Shi et al., use an artificial neural network as a classifier and shows that that ANN performs better in Steganalysis than Bayes classifier due to its powerful learning capability.

Pseudo random number generator is another approach for neural networks in cryptography. Cheng and Chan [54], state that randomness increases the security of the cryp-

tosystem. In their work, they develop a pseudo-random number generator which proves to be very suitable for practical implementation on efficient flow encryption cryptosystems. The pseudo-random number generator takes advantage of multilayer perceptron neural networks. In the overfitting process, the network will not be able to predict the input pattern when it receives unknown input patterns and will give unpredictable results [55]. In [50], Karras and Zorkadis state that a multilayer neural network can be used as an independent random generator and also as a method to strengthen existing generators. This process is performed by taking the pseudo-random numbers generated by linear computational generators as input to the neural networks.

Chapter 5

Methodology

This chapter describes the methodology used to implement the ANN-based asymmetric cryptography system. The development of this system focuses on providing better computational time performance for the data encryption and decryption process, implementing an alternative to the traditional algorithms used in information security.

5.1 Research Approach

In this work, an experimental study was developed to design an asymmetric key cryptographic system based on ANN, capable of providing an adequate level of security and low execution time. This study was carried out in three parts: (1) Autoencoder neural network calibration to randomize the training process and generate a different ASCII code codification according to an initialization password. (2) Calibration of the neural weights that will work as a public and private key. (3) Comparative study of system performance and security analysis to determine the system resistance to attacks by hackers. This work was developed in Borland C++ and uses an autoencoder neural network architecture to generate a public and private keys. Additionally, an algorithm was designed to increase the randomness of the network, both in the initialization of the synaptic weights and in the network training processes.

5.2 Autoencoder Neural Network Architecture.

To develop this system, an autoencoder with 8 neurons in the input layer, 10 neurons in the hidden layer and 8 neurons in the output layer was selected. This architecture was selected taking into account the following factors: number of preset characters in the complete ASCII code, size of the generated keys and the resizing of plain text. The graphical representation of the autoencoder is shown in Figure 5.1.

The full ASCII code was encoded in 2^8 bits, this means that each character used in the cryptosystem had an 8-bit representation. Each 8-bit set was used as input to the neural network. Due to this, the input and output layers were made up of eight neurons (following the symmetry of the autoencoder). The size of the neural network allowed the system to be simple but robust. The ten neurons that were established in the intermediate layer allowed the generation of keys of considerable length. This implies that

trying to “emulate” a key using a brute force approach involves a great computational cost. Furthermore, having ten neurons in the hidden layer implied that the ASCII code suffered from a considerable resizing, which means that the security level of the system increased.

The selected activation function was the sigmoid function. See Equation 5.1.

$$f(x) = \frac{1}{1 + e^{-\beta x}}, \quad (5.1)$$

where β is the gain value.

Due to the gain β was set as a variable value, each layer of the neural network has a “different” sigmoid function. The hidden layer uses a β value equal to 10.5, while the output layer has a β value equal to 5.5. This allowed the network weights W_E and W_D to have different ranges for their possible values at the end of the training.

The general ranges are:

$$W_E \in (-0.75, 0.75)$$

$$W_D \in (-2, 2)$$

However, these may vary slightly.

The last parameter of the network to be established was the learning rate with a value of 0.25. All network parameters were established after several coding tests performed with the proposed network architecture.

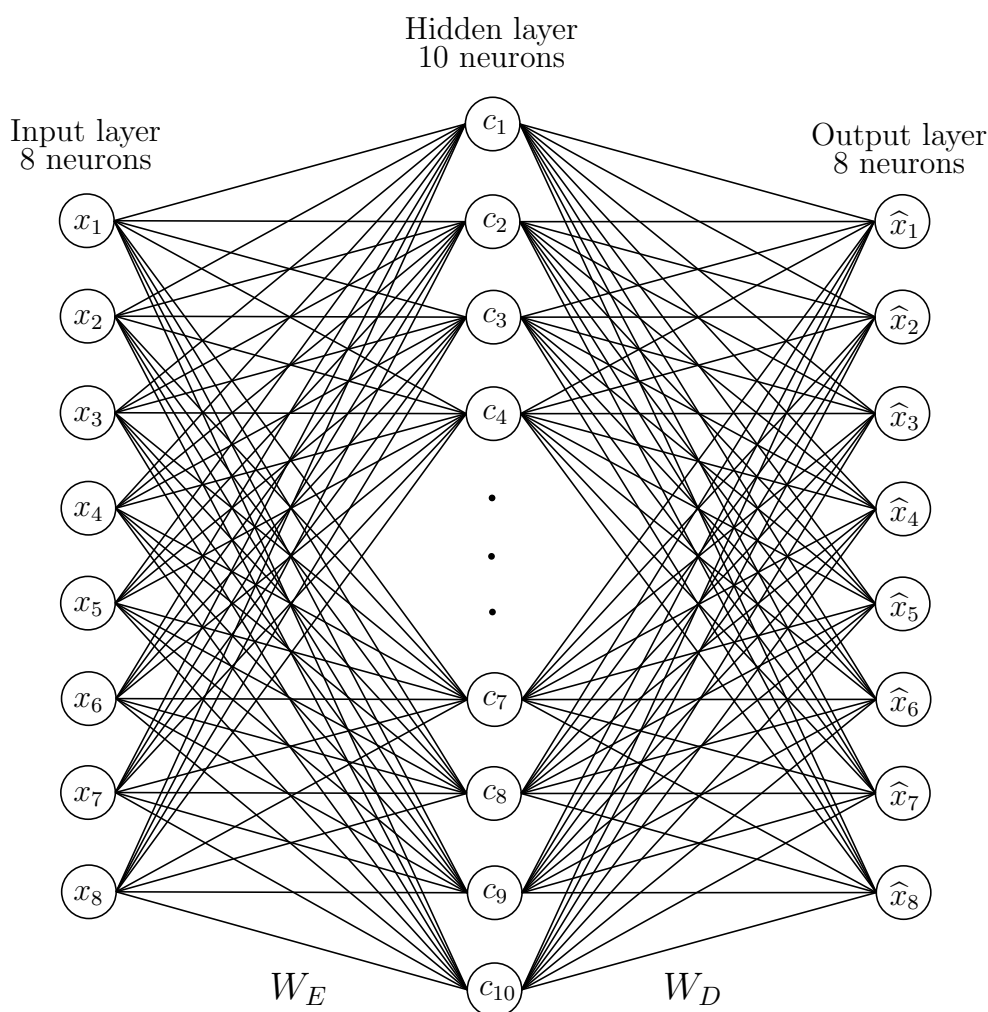


Figure 5.1: Autoencoder architecture used in the proposed system. This architecture is composed of 8 neurons in both the input and output layers (following the definition of autoencoder) and the hidden layer contains 10 neurons.

5.3 Randomization Algorithm

The proposed system implemented a randomization algorithm that allowed generating initial random synaptic weights and a vector of random indices. This algorithm was based on changing “seeds” in conjunction with the *rand()* function provided by Borland C ++. The change of seeds was an important process to increase the randomness in the system since we avoid using recursive processes on the same seed.

Due to, the final synaptic weights of the neural network depend on the initial random values and how the input data are entered. The proposed algorithm generates different keys (W_E and W_D) for each initialization password. A description of the randomization algorithm is shown in Figure 5.2.

The algorithm used two alphanumeric strings as input. The first string is entered by the system user and is called the initialization password. The second string is a random sequence of characters, which was defined exclusively for the cryptosystem. The two

chains were subjected to a loop of XOR operations between all the elements that made up the chains. This generated several different seeds (65,536 possible seeds) which were used in the Borland C ++ random function to generate pseudo-random numbers. When a pseudo-random number between $[-0.5,0.5]$ is generated with a seed X , this is placed as initial synaptic weight value, and then the seed changes. Each seed is used only once in the whole process since the seed value change once the pseudo-random number is generated. When a pseudo-random number between $[-0.5, 0.5]$ was generated with a seed X , this was established as the initial synaptic weight value. Each seed was used only once in the whole process, since the value of the seed changed once the pseudo-random number was generated. In addition, the random number generation algorithm was implemented to create a pseudo-ordered index vector. This vector was used to reorder the training data entered into the neural network. A vector of pseudo-ordered indexes was created to avoid a high computational time at the moment of key generation. This algorithm increased the randomness in the process of initialization and training of the neural network. Therefore, the randomness in the entire cryptographic system increased, especially in the generation of the private key.

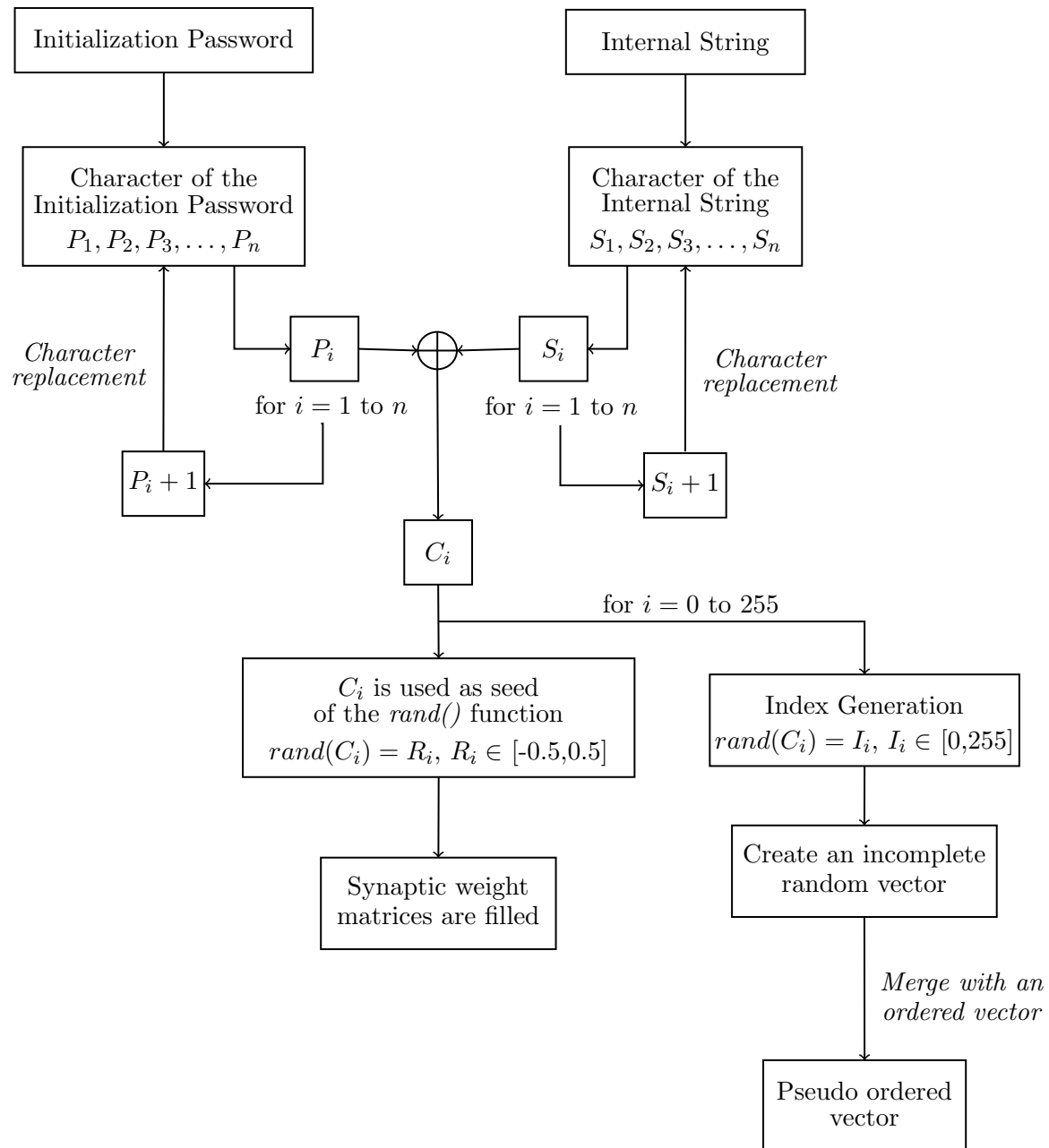


Figure 5.2: A representation of the randomization algorithm that shows the entire process that the initialization password undergoes. The algorithm is used to obtain several seeds with which the weight matrix of the neural network is initialized.

5.4 Asymmetric Key Cryptography Model

The system implemented an autoencoder neural network architecture to generate a public and private key. These keys correspond to the synaptic weights of the network (W_E and W_D). The system have a randomization algorithm that depends on a secret initialization password entered by the user and a random character string specific for this system.

In addition, the system performed a preprocessing of the information before generating

the ciphertext from the plain text. In the same way, to generate the plain text from the ciphertext, a preprocess was carried out inverse to the one initially performed. A complete scheme of the system is presented in Figure 5.3.

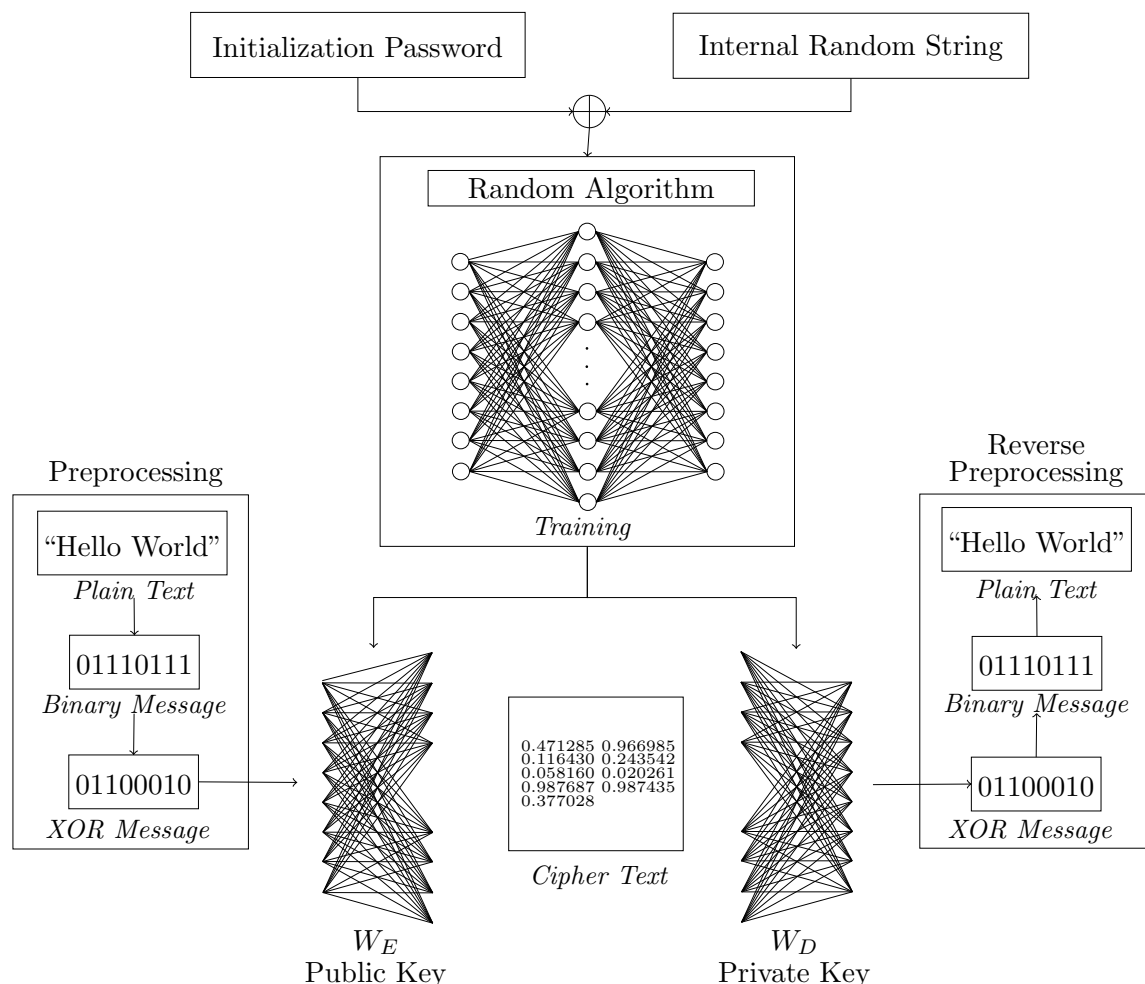


Figure 5.3: Scheme of the proposed cryptographic system which is composed of the initialization of the system, the creation of the keys and the data encryption/decryption process.

5.5 Encryption and Decryption Process

This section describes in detail a message encryption and decryption process.

5.5.1 Encryption

A flow chart of the encryption process is presented in Figure 5.5. To perform data encryption, the plain text must be pre-processed. First, the original message was translated into ASCII code, thus obtaining a binary text. Next, the binary text was divided into 8-bit sets and all of these were subjected to various XOR operations (see Figure 5.4). When the “XOR text” was obtained, it was encrypted using the public key, obtaining

the ciphertext. The resulting ciphertext was resized from “XOR text”, and each 8-bit set was converted to a set of ten floating numbers. An example of message encryption is presented in Table 5.1.

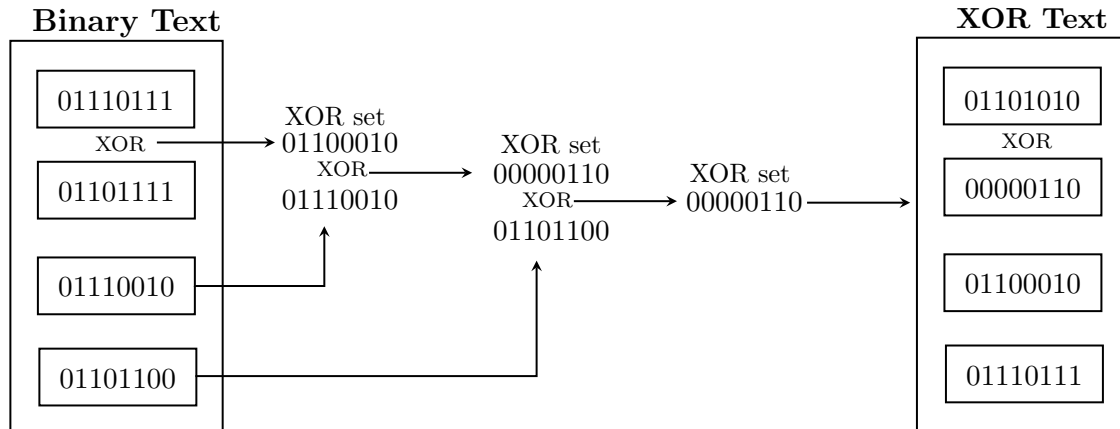


Figure 5.4: XOR text generation corresponding to the information preprocessing that the system performs.

Plain Text	Binary Message	XOR Message
"World"	011101110110111 101110010011011 0001100100	011000100000011 001101010000110 0001110111
Ciphertext: 0.471285 0.966985 0.116430 0.243542 0.058160 0.020261 0.987687 0.987435 0.377028 0.002005 0.229696 0.037587 0.990485 0.025279 0.470377 0.982231 0.980661 0.643427 0.011464 0.004897 0.554947 0.958932 0.923121 0.307470 0.193312 0.000062 0.983166 0.925149 0.204126 0.022497 0.985374 0.610509 0.876300 0.158316 0.794756 0.022864 0.251481 0.068353 0.012083 0.998407 0.975261 0.653203 0.107748 0.082523 0.000600 0.980367 0.991929 0.711553 0.053618 0.007623		

Table 5.1: Example of message encryption

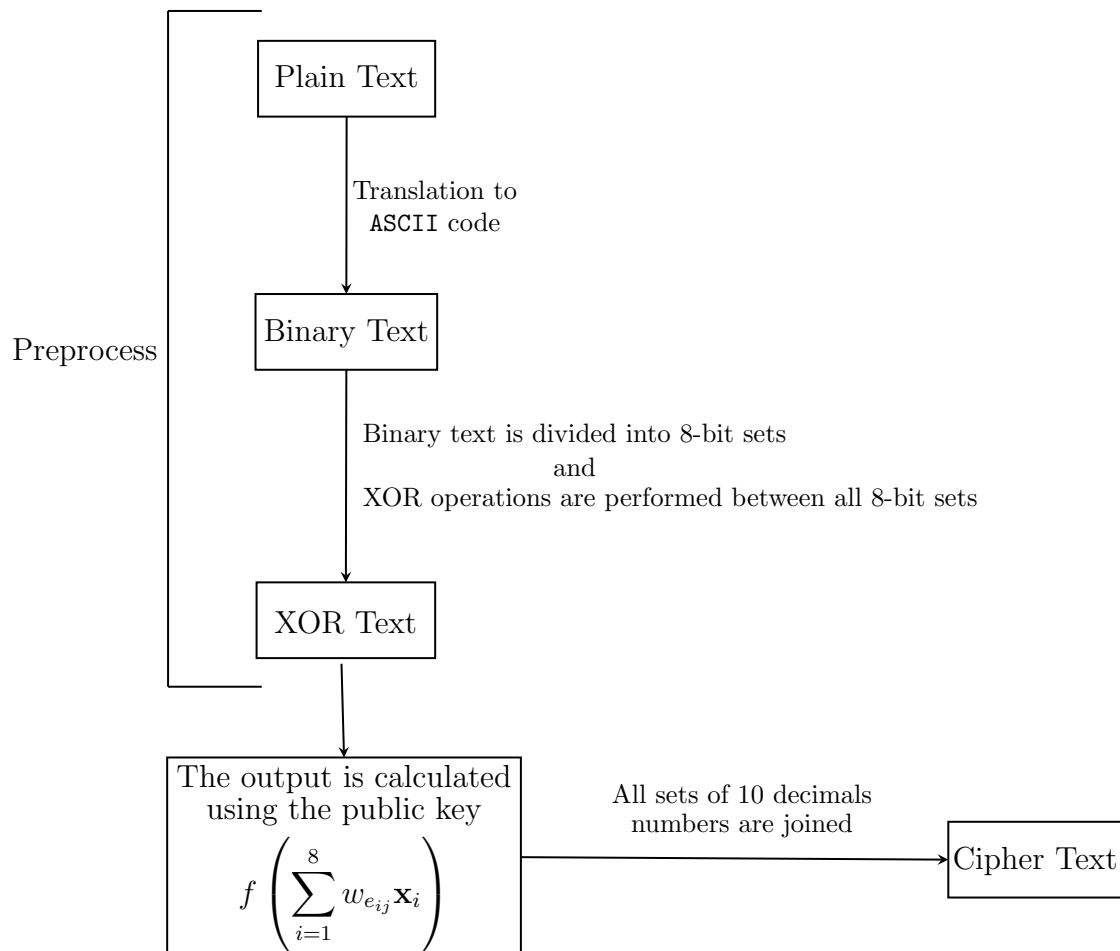


Figure 5.5: Flow chart that shows how the ciphertext is obtained from the plain text. The diagram shows the information preprocessing and the use of the public key in the encryption process.

5.5.2 Decryption

A flow chart of the decryption process is presented in Figure 5.6. In the decryption process, the ciphertext was divided into blocks of ten floating numbers and, using the private key with the Equation 5.2 the “XOR text” was obtained. From this point, a reverse process to the preprocessing was performed (section 5.5.1). The “XOR text” was divided into 8-bit blocks and was subjected to a recovery process by means of XOR operations. Then the binary text was obtained and later the plain text. An example of message decryption is presented in Table 5.2.

$$\Theta(x) \begin{cases} 1, & \text{if } x \geq 0.5 \\ 0, & \text{if } x < 0.5 \end{cases} \quad (5.2)$$

Ciphertext: 0.471285 0.966985 0.116430 0.243542 0.058160 0.020261 0.987687 0.987435 0.377028 0.002005 0.229696 0.037587 0.990485 0.025279 0.470377 0.982231 0.980661 0.643427 0.011464 0.004897 0.554947 0.958932 0.923121 0.307470 0.193312 0.000062 0.983166 0.925149 0.204126 0.022497 0.985374 0.610509 0.876300 0.158316 0.794756 0.022864 0.251481 0.068353 0.012083 0.998407 0.975261 0.653203 0.107748 0.082523 0.000600 0.980367 0.991929 0.711553 0.053618 0.007623		
XOR Message	Binary Message	Plain Text
011000100000011 001101010000110 0001110111	011101110110111 101110010011011 0001100100	"World"

Table 5.2: Example of message decryption

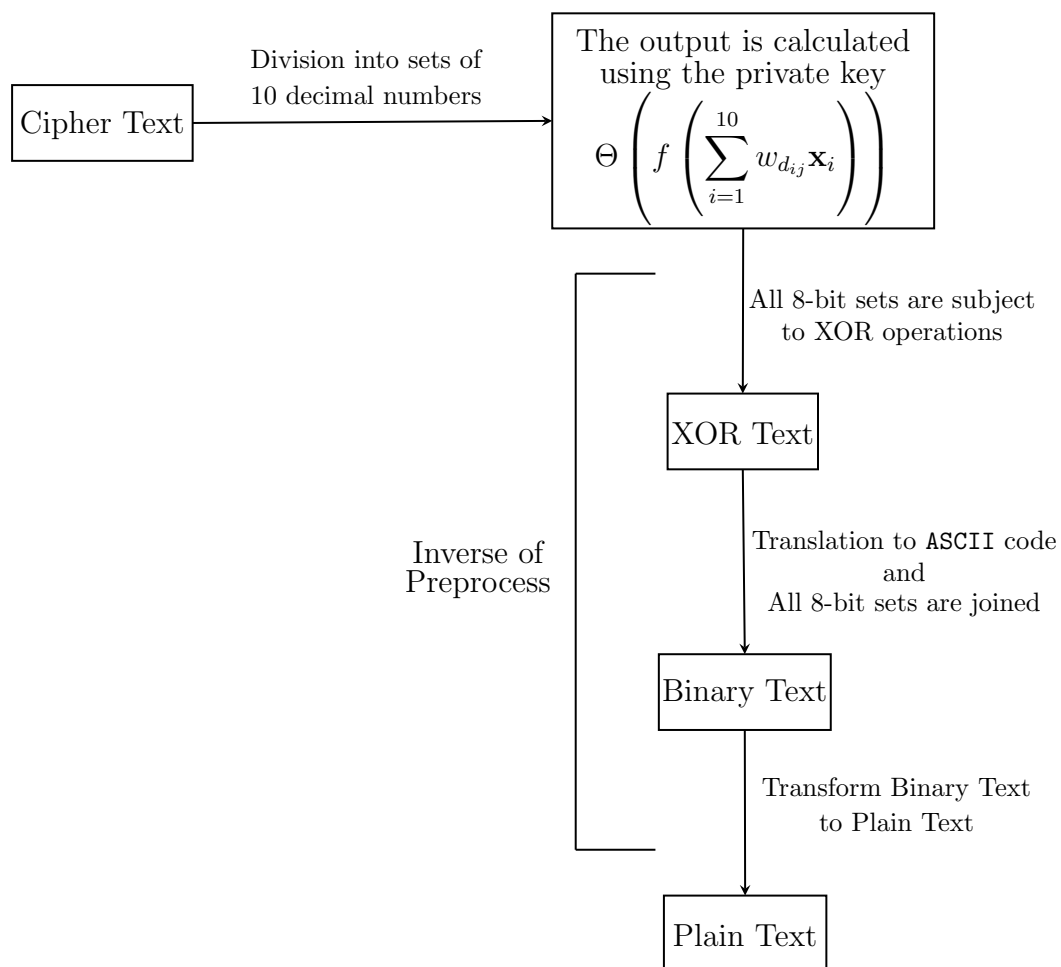


Figure 5.6: Flow chart showing how to get plain text from ciphertext. The diagram includes the inverse of preprocessing and how the private key is used in the decryption process.

Chapter 6

Performance and Security Parameters

This chapter describes the parameters used to evaluate the performance and security of the proposed cryptosystem. Different parameters are taken into account to evaluate performance, such as encryption time, decryption time and key generation time. These parameters are compared with data from asymmetric cryptographic algorithms such as RSA, ECC, Paillier and Elgamal. In addition, the security parameters considered are precision, network tolerance, private key, determination of significant digits and neural network security. Finally, a way of measuring the efficiency of the randomization algorithm in the creation of keys is established.

6.1 Hardware Characteristics

The characteristics of hardware used in the cryptographic system performance evaluation are taken into account to establish that the data obtained is subject to the used hardware. The system is tested under a Windows 10 64-bit operating system, 8Gb RAM and Intel(R) Core(TM) i7-5500U CPU 2.40GHz 2.40GHz.

6.2 Performance Parameters

6.2.1 Encryption Time

Encryption time is determined based on the time required by an algorithm or system to translate a plain text into a ciphertext. The encryption time depends only on the size of the plain text and this is measured in seconds.

6.2.2 Decryption Time

Decryption time is determined during the translation into the plain text from the ciphertext. For this work, it is preferred that decryption and encryption time will be similar to guarantee the efficiency of the cryptographic system.

6.2.3 Key Generation Time

The time required to generate the keys is equivalent to the training time of the artificial neural network. It depends on the size of the string needed to initialize the neural network training.

6.3 Security Parameters

6.3.1 Precision

Precision is given by the number of characters successfully retrieved, over the total number of characters that the complete ASCII code has. This value is measured in percentages and it can change during the training process or due to a variation in the weights in a training network (public and private keys).

6.3.2 Network Tolerance

Network tolerance refers to the amount of noise that the synaptic weights of trained network support before its data recovery fail. The noise that enters the network alters the number of significant digits that each synaptic weight needs in data recovery. In other words, the network needs less significant digits per weight to recover the data with high precision.

6.3.3 Private Key Security

The private key is a portion of all synaptic weights involved in data recovery. The synaptic weights of an ANN are continuous values, generally with a large number of significant digits. Therefore, to hack a private key, almost an infinite number of attempts are needed to get the exact numbers. Equation 6.1 is used to determine the number of attempt necessary to hack the private key.

$$VR_n^m = n^m. \quad (6.1)$$

Where n is the set of all possible elements that can be used to generate the public key. In this case, n depends on the number of significant digits and the rank corresponding to the synaptic weights of a network with a high precision. Besides m is a subset of n , that depends on the number of elements in the private key (size of W_D).

6.3.4 Significant Digits Determination

To identify the minimum number of significant digits that the public key use to recover the data, a randomly float values in the range [0.001-0.1] are added and subtracted to the public key. When the precision decreases more than 75%, the number of significant digits of the float value capable to have a high accuracy is established as the minimum amount of significant digits.

6.3.5 Neural Network Security

The generation of public and private keys depends on two character strings. The first is a user-entered string, and the second is an internal random character string. The randomization algorithm uses the two strings to initialize the synaptic weights of the network and generate a chain of pseudo-ordered indexes, which will be used in training. Therefore, in order to obtain a neural network capable of generating identical keys, a hacker would need to guess by brute force the password entered by the user.

So, the hacker has to guess the size of the string and the exact combination of characters that were used. Equation 6.2 is used to calculate the time necessary to hack the network.

$$T = (N^M) \times t_{nn}. \quad (6.2)$$

Where N is the total number of printable ASCII characters, M is the length of the string entered, and t_{nn} is the neural network training time.

6.3.6 Randomization Algorithm Efficiency

The randomization algorithm depends mainly on an initialization password. If the password changes, the resulting keys must be totally different. To measure the efficiency in the randomization of data carried out by the algorithm, it is proposed to compare the keys generated by two initialization passwords with high similarity.

Chapter 7

Results

This chapter presents the results of the performance evaluation and the security analysis of the proposed system including the randomization. The encryption and decryption time was compared with the results obtained by Maqsood [3], Matta [56] and Farah [57].

7.1 Performance Evaluation

To evaluate the performance of the cryptographic system, the encryption/decryption time and the key generation time were used.

7.1.1 Encryption and Decryption Time of Cryptosystem

To determine the overall performance with respect to encryption and decryption time, the proposed system ran with the following input file sizes: 200 KB, 300 KB, 400 KB, 500 KB, 600 KB. For each file size, the cryptosystem was run ten times to have a representative average time. The results are shown in Figure 7.1.

The results showed that when the file size increases, the time that the cryptosystem requires to encrypt and decrypt information also increases. This was because the cryptosystem performs a resizing of the data, this means that the size of the file that is decrypted is greater than the one that was encrypted. Additionally, it was obtained that the encryption time is less than the decryption time. However, the time difference between each process is small, and in some cases the decryption time became equal to the encryption time. This was because the encryption and decryption are done through matrix operations that do not represent a high computational cost. This produces a similarity in the times of each process and a reduced computational time.

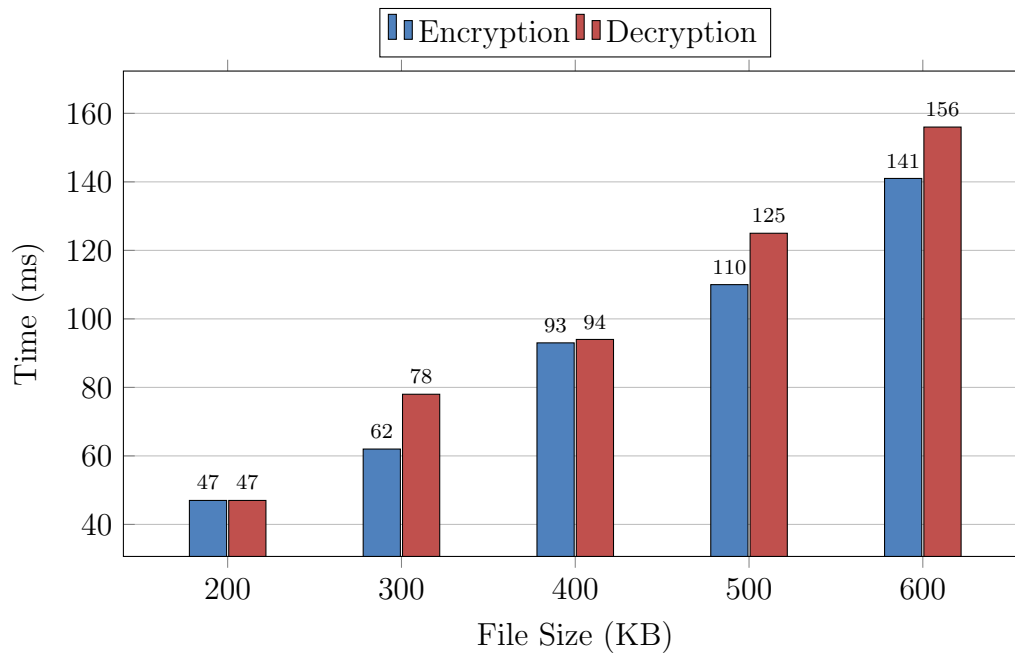


Figure 7.1: Execution time corresponding to the encryption and decryption processes of the cryptographic system.

7.1.2 Key Generation Time of the Cryptosystem

To determine the average time that the cryptosystem needs to generate the public and private key, it was executed with the following initialization password lengths (numbers of characters): 4, 5, 10, 12, 14, 15. For each password of initialization, the system was executed ten times and the average value was taken. Table 7.1 shows the average times for each initialization password length and the sizes of the generated keys. In Figure 7.2, the average times of each password length are shown graphically, and the key creation time does not depend on the password length. Additionally, it was determined that the size of the keys is constant and does not depend on the size of the initialization password. Since the generation time of the keys does not depend on the length of the initialization password, a general average of 27.47 seconds was determined. In addition, the length of the password is always the same, so it was not taken into account for future calculations.

Table 7.1: Key Generation Time and Keys Size of the Cryptosystem

Initialization Password Length	Time (s)	Public Key Size (KB)	Private Key Size (KB)
4	26.4	1	1
5	32.7	1	1
10	22.39	1	1
12	30.9	1	1
14	21.03	1	1
15	31.4	1	1
Average:	27.47	1	1

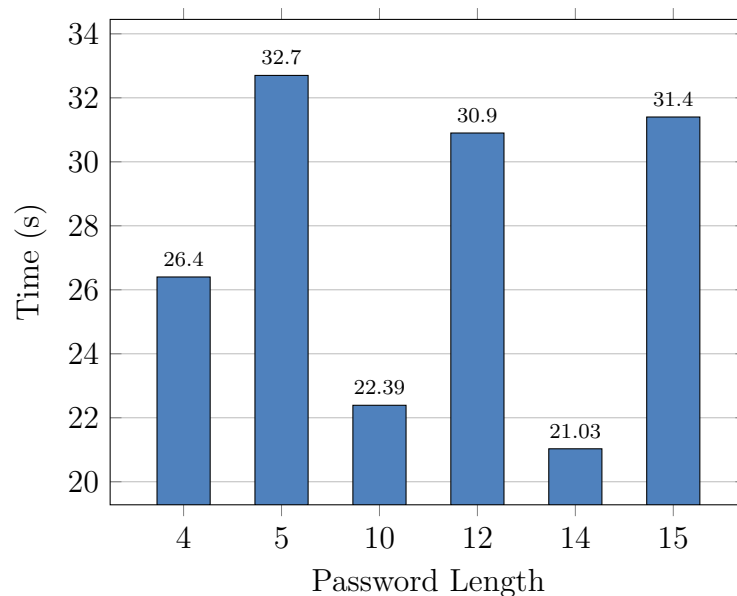


Figure 7.2: Time required by the cryptographic system to generate a set of keys with respect to the length of the initialization password.

7.1.3 Comparison between RSA, ElGamal and the proposed cryptosystem

Maqsood et al. [3] analyzed the performance of RSA and Elgamal asymmetric cryptographic algorithms in terms of encryption/decryption time and key generation time. The algorithms were implemented in Java (Eclipse platform version: 3.3.1.1) and the experiments were performed under Intel Pentium processor with 2.34 GHz and 1 GB of memory. The reported performance was measured in seconds and was based on the execution of the algorithms with different text file sizes such as 32 KB, 126 KB, 200 KB, 246 KB and 280 KB.

To compare the proposed cryptosystem with the RSA and ElGamal algorithms, data files of the same size as those used in the comparison made by Maqsood were created.

Encryption Time

The comparison of encryption times between RSA, ElGamal and the proposed cryptosystem are shown in Table 7.2 and Figure 7.3. The encryption times reported for the cryptosystem are much lower than those of RSA and ElGamal. Additionally, it is seen that as the file size increases, the cryptosystem requires more time to perform the encryption.

Table 7.2: Encryption Time comparative with Maqsood et al.

File size (KB)	RSA Time (ms)	ElGamal Time (ms)	Cryptosystem Time (ms)
32	130	450	10
126	520	1030	31
200	740	1410	47
246	1110	1750	47
280	1390	1830	62

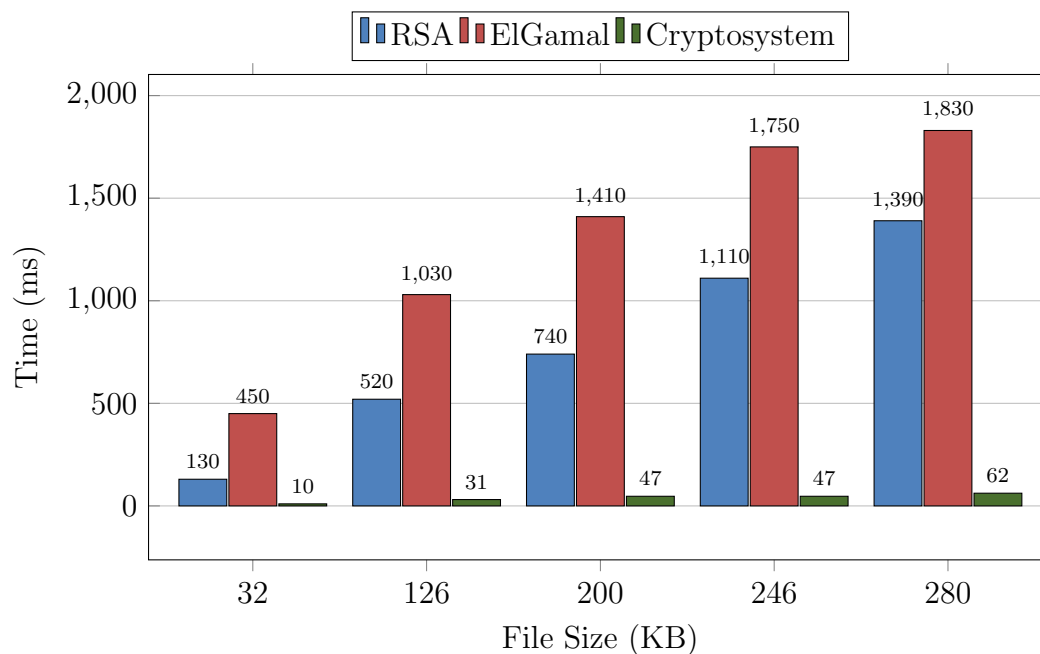


Figure 7.3: Encryption times of the RSA, ElGamal algorithms and the proposed cryptosystem. The encryption times of RSA and ElGamal were obtained by Maqsood et al.

Decryption Time

The decryption time comparison between the RSA, ElGamal algorithms and the proposed cryptosystem are shown in Table 7.3 and Figure 7.4. The behavior was similar to the encryption process where the decryption times are much lower than those of SRA and ElGamal and if the file size increases, so does the decryption time. Additionally, the

cryptosystem decryption times are very similar to the encryption times. This was because the decryption performed by RSA and ElGamal is based on mathematical operations between very large prime numbers. While the cryptosystem based on matrix operations avoids the use of large prime numbers that can increase the computational time.

Table 7.3: Decryption Time comparative with Maqsood et al.

File size (KB)	RSA Time (ms)	ElGamal Time(ms)	Cryptosystem Time (ms)
32	150	430	16
126	430	850	32
200	660	1300	47
246	930	1640	62
280	230	1640	63

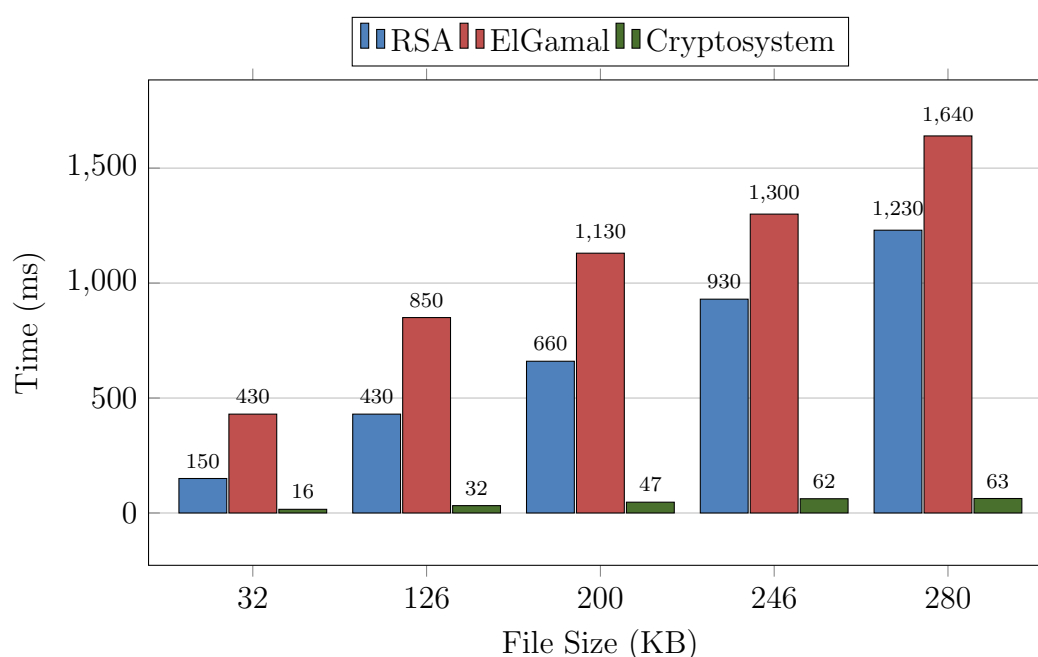


Figure 7.4: Decryption times of the RSA, ElGamal algorithms and the proposed cryptosystem. The decryption times for RSA and ElGamal were obtained by Maqsood et al.

Key Size and Generation Time

The size of the keys and key generation time are shown Table 7.4. The time that the cryptosystem needs to generate a set of keys is much longer than that reported for RSA and ElGamal. To determine the total size of the set of keys, the size of the encryption key was added to the size of the decryption key. The size of the keys generated by the proposed cryptosystem is much larger than the key sizes for RSA and ElGamal. The comparison of the generation times is shown in Figure 7.5 and the comparison of the key sizes is shown in Figure 7.6.

Table 7.4: Key Generation Time comparative with Maqsood et al.

	Key Size (bits)	Generation Time (s)
RSA	1024	0.287
ElGamal	160	0.86
Cryptosystem	16000	27.74

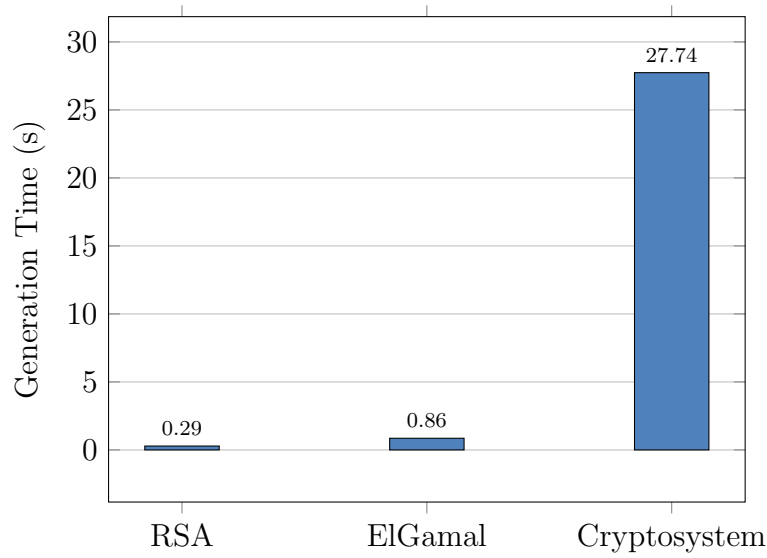


Figure 7.5: Comparison of key generation times between RSA, ElGamal and the proposed Cryptosystem. The times of RSA and ElGamal were obtained by Maqsood et al.

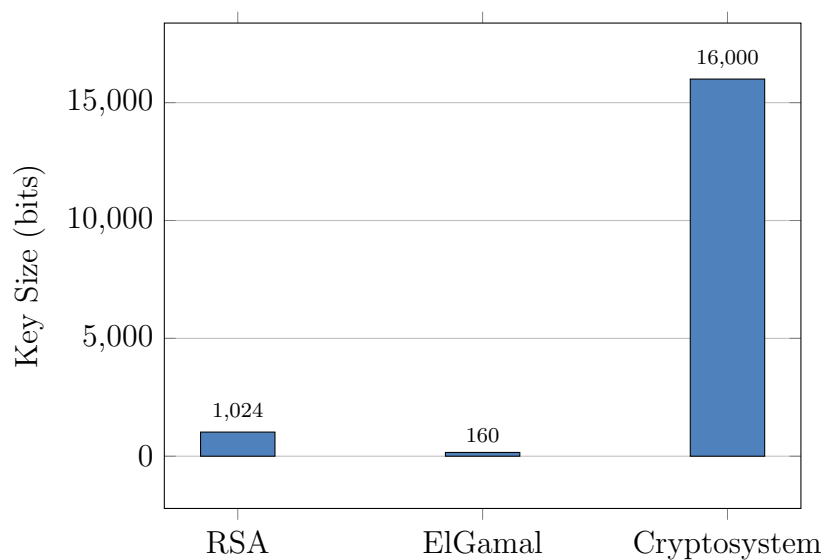


Figure 7.6: Size comparison of the keys generated by RSA, ElGamal and the proposed Cryptosystem. RSA and ElGamal key sizes were obtained by Maqsood et al.

7.1.4 Comparison between RSA-16bits, ECC-16bits and the proposed cryptosystem

Matta et al [56] carried out the performance analysis of RSA-16bits and ECC-16bits encryption algorithms with a steganographic approach. Even though Matta's work uses steganography rather than an encryption approach where the plain text is transformed to a ciphertext, it provides useful information for evaluating the proposed system. The RSA-16bits and ECC-16bits algorithms were evaluated separately and for each one, the key generation time and the encryption/decryption times were analyzed. Matta, evaluate RSA-16bits and ECC-16bits with the follow file sizes: 22 KB, 87 KB and 174 KB.

Encryption Time

To compare the proposed cryptosystem with the RSA-16bits and ECC-16bits algorithms, data files of the same size as those used in the comparison made by Matta [56]. The comparison of the encryption times between RSA-16bits and ECC-16bits and the proposed cryptosystem are shown in Table 7.5 and Figure 7.7. The encryption times presented by the cryptosystem are much lower than those of RSA-16bits and ECC-16bits. This is because the proposed cryptosystem is based on matrix operations that require a reduced computing time. In addition, the cryptographic algorithms that are used for the purchase have a serial execution approach based on operations with very large prime numbers, for which they require a high computation time.

Table 7.5: Encryption Time comparative with Matta et al.

File size (KB)	RSA-16. Time (ms)	ECC-16. Time (ms)	Cryptosystem Time (ms)
22	3782	51391	16
87	4297	93741	16
174	4265	113947	32

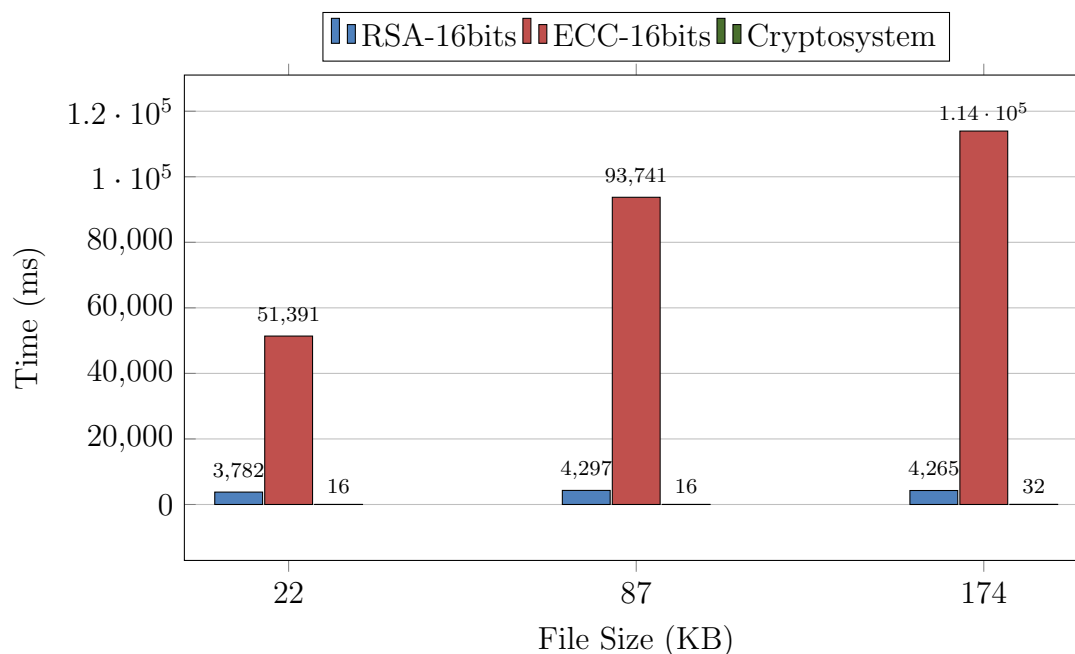


Figure 7.7: Encryption times of RSA-16bits, ECC-16bits and the proposed cryptosystem. The encryption times of RSA and ECC were obtained by Matta et al.

Decryption Time

The comparison of the decryption times between RSA-16bits and ECC-16bits and the proposed cryptosystem are shown in Table 7.6 and Figure 7.8. The decryption times presented by the cryptosystem are much lower than the times of RSA-16bits and ECC-16bits. This is because one uses matrix operations and the other uses operations and functions that involve very large prime numbers. When comparing the data presented in Figure 7.7 and Figure 7.8, the cryptosystem requires more time in the decryption process than in the encryption process. This is due to the re-dimensioning that plaintext undergoes when it is encrypted. Ciphertext is larger than plain text and for that reason decryption takes more time than encryption.

Table 7.6: Decryption Time comparative with Matta et al.

File size (KB)	RSA-16. Time (ms)	ECC-16. Time (ms)	Cryptosystem Time (ms)
22	4328	24187	16
87	4188	75402	31
174	4390	115137	47

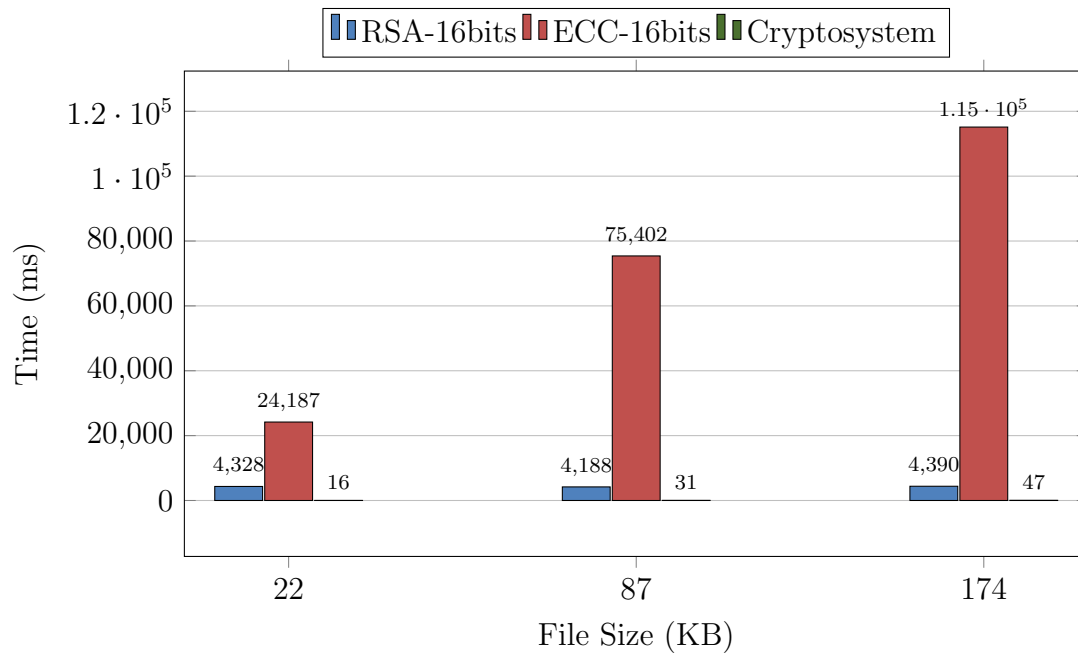


Figure 7.8: Decryption times of RSA-16bits, ECC-16bist and the proposed cryptosystem. The decryption times of RSA and ECC were obtained by Matta et al.

Key Generation Time

The key generation times are shown in Table 7.7 and Figure 7.9. To obtain the generation time of the set of keys, the creation time of the public key plus the creation time of the private key was added. In the case of the cryptosystem, it only has a single creation time for the two keys. The key generation time required by the proposed cryptosystem is much higher than the declared time for RSA-16bits, however it presents a much lower time than that required by ECC-16bits.

Table 7.7: Key Generation Time comparative with Matta et al.

	Public Key	Private Key	Total Time (s)
RSA-16bits	0.031	0.015	0.046
ECC-16bits	123.203	122.922	246.125
Cryptosystem	-	-	27.47

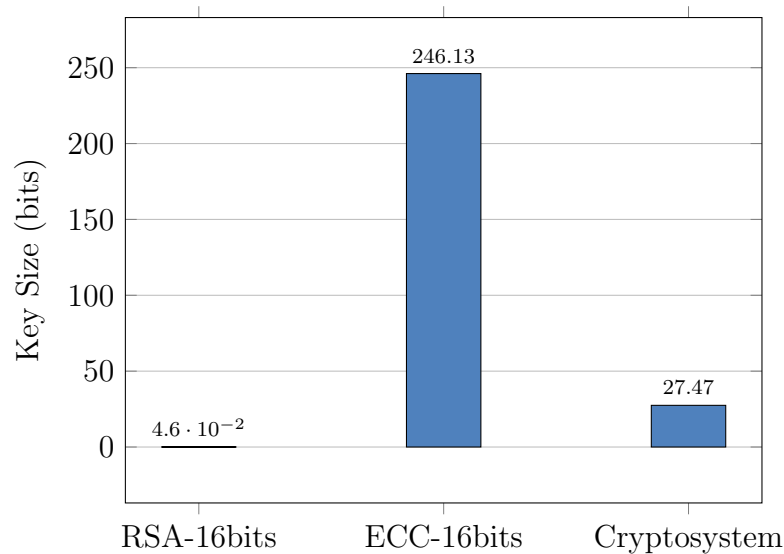


Figure 7.9: Comparison of key generation times between RSA-16bits, ECC-16bits and the proposed Cryptosystem. The times of RSA and ECC were obtained by Matta et al.

7.1.5 Comparison between RSA, ElGamal, Paillier and the proposed cryptosystem

Farah et al. [57] analyzed the performance evaluation of the RSA, ElGamal and Paillier algorithms, comparing the encryption and decryption time of each algorithm. It was performed under Windows XP operating system with Intel (R) Core (TM) 2 Duo CPU 2.09 GHz and 4 GB of RAM. The comparison was measured in seconds and use a different file sizes such as: 68 KB, 105 KB, 124 KB and 235 KB.

Encryption Time

To compare the proposed cryptosystem with the RSA, ElGamal and Paillier algorithms, data files of the same size as those used in the comparison made by Farah [57]. The comparison of the encryption times between RSA, ElGamal, Paillier and the proposed cryptosystem are shown in Table 7.8 and Figure 7.10. In relation to all the algorithms mentioned above, the proposed cryptosystem presents much lower computation times. This comparison verifies that the cryptosystem based on matrix operations has lower computational times than algorithms based on serial operations that involve large prime numbers.

Table 7.8: Encryption Time comparative with Farah et al.

File size (KB)	RSA (ms)	ElGamal (ms)	Paillier (ms)	Cryptosystem (ms)
68	490	900	470	16
105	530	1400	500	31
124	900	50	500	31
235	1500	3000	2520	47

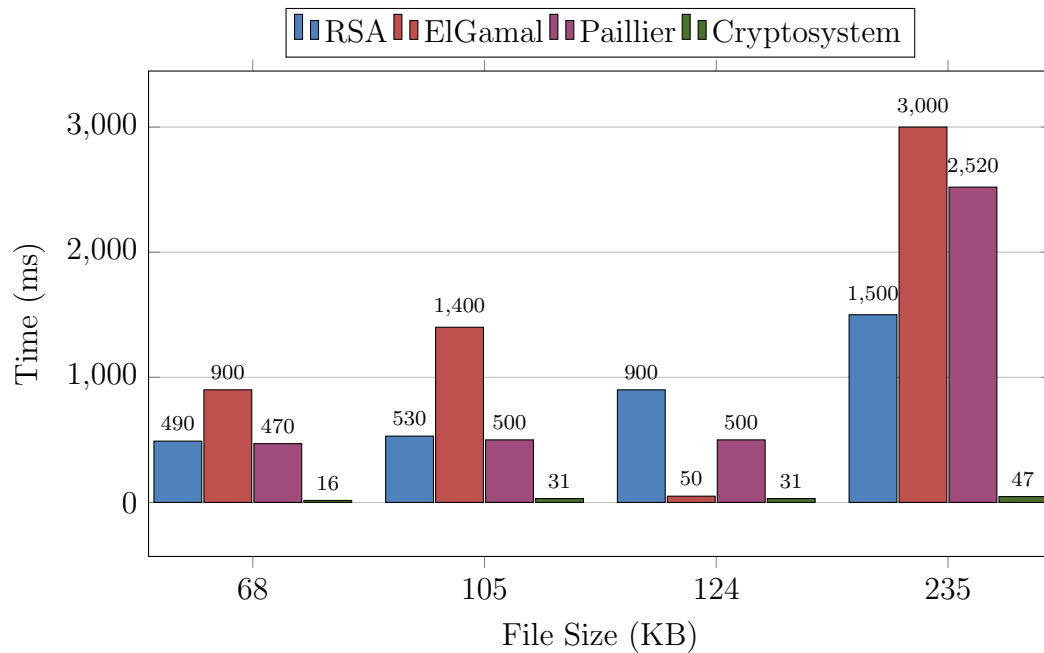


Figure 7.10: Encryption times of RSA, ElGamal, Paillier and the proposed cryptosystem. The encryption times of RSA, ElGamal and Paillier were obtained by Farah et al.

Decryption Time

Comparison of decryption times are shown in Table 7.9 and Figure 7.11. The proposed cryptosystem presented lower decryption times than the RSA, ElGamal and Paillier algorithms. When comparing Figure 7.10 with Figure 7.11, it can be seen that the cryptosystem requires more time to perform the decryption than the encryption. It can be seen in Figure 7.11 that ElGamal has low computation times, however these are still high in relation to those obtained by the cryptosystem

Table 7.9: Decryption Time comparative with Farah et al.

File size (KB)	RSA (ms)	ElGamal (ms)	Paillier (ms)	Cryptosystem (ms)
68	30000	500	7500	16
105	50000	1500	8200	31
124	59000	2300	14800	32
235	95000	3200	36000	63

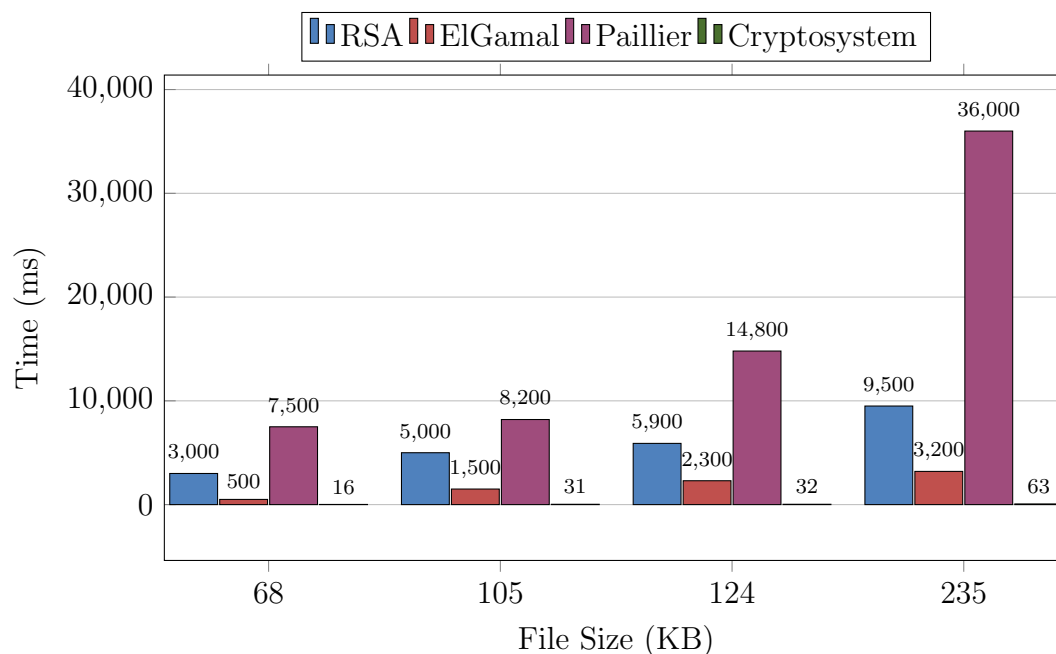


Figure 7.11: Decryption times of RSA, ElGamal, Paillier and the proposed cryptosystem. The decryption times of RSA, ElGamal and Paillier were obtained by Farah et al.

7.2 Security Evaluation

To determine the security of the private key, the parameters and equations defined in the section 6.3 are taken into account.

7.2.1 Private Key Security Results

Table 7.10: Network Tolerance

Thousandths (4 digits)		Hundredths (3 digits)		Tenths (2 digits)	
Noise	Accuracy	Noise	Accuracy	Noise	Accuracy
0,001	99,61	0,01	100,00	0,1	66,67
0,002	99,61	0,02	100,00	0,2	17,65
0,003	99,61	0,03	99,22	0,3	3,92
0,004	99,61	0,04	98,43	0,4	0,78
0,005	100,00	0,05	98,04	0,5	0,00
0,006	100,00	0,06	92,55	0,6	0,00
0,007	100,00	0,07	86,67	0,7	0,00
0,008	99,61	0,08	78,43	0,8	0,00
0,009	99,61	0,09	74,12	0,9	0,00

The results in Table 7.10 shows that the number of significant digits necessary for the cryptosystem to have high precision is in the order of hundreds. This means that the

numbers used in the cryptosystem must be composed of at least 3 digits. The rank of the private key is (-2,2) (See Section 5.2) and the size of W_D is 80. According to the rank and the significant digits, we have the set $[-1.99, \dots, 1.99]$, that contains 399 elements. Using Equation 6.1, the number of attempts a hacker needs to replicate the correct sequence of numbers that make up the private key is calculated.

$$VR_{399}^{80} = 399^{80} \longrightarrow VR_{399}^{80} = 1.19628 \times 10^{288}(\text{attempts})$$

To calculate the time to perform all this attempts, an hypothetical machine capable to realize 1×10^7 attempts per second is assumed.

$$x(\text{years}) = \frac{1.19628 \times 10^{288}(\text{attempts}_{total}) \times 1(\text{seconds}) \times 1(\text{years})}{1 \times 10^7(\text{attempts}_{machine}) \times 3.15 \times 10^{193}(\text{seconds})} \quad (7.1)$$

Evaluating Equation 7.1 we obtain that takes $3,79 \times 10^{193}$ years to hack the private key.

7.2.2 Neural Network Security

To obtain the correct password capable to generate the correct public and private key, the hacker needs to know the system initialization password. We know that the the average training time for the network is 27.47 seconds (see Table 7.1). The total number of printable characters of the ASCII code are 223. Then using Equation 6.2 we have the following results.

Table 7.11: Neural Network Security Results

Initialization Password Length	Equation	Time (s)	Time (years)
4	$T = (223^4) \times 27.47(s)$	67932580424	2,15E+03
5	$T = (223^5) \times 27.47(s)$	15148965434612	4,80E+05
10	$T = (223^{10}) \times 27.47(s)$	8,35425E+24	2,65E+17
12	$T = (223^{12}) \times 27.47(s)$	4,15448E+29	1,32E+22
14	$T = (223^{14}) \times 27.47(s)$	2,06598E+34	6,55E+26
15	$T = (223^{15}) \times 27.47(s)$	4,60714E+36	1,46E+29

Table 7.11 shows the time in years that it would take to hack the cryptosystem in relation to the length of the initialization password used. The computation time required to hack the system with a brute force approach increases exponentially when the length of the initialization password or increases. All the times presented in Table 7.11 are extremely large which makes the hacking process computationally unfeasible.

7.2.3 Randomization Algorithm Performance

To determine the performance of the randomization algorithm, keys are created with the initialization passwords “helloworld” and “helloworle”. The difference between the passwords is minimal to determine the effectiveness of the algorithm.

Pseudo-ordered Index Vector

Figure 7.12 shows the distribution of the pseudo-ordered vectors of indices created with the passwords “helloworld” and “helloworle”. The distribution of the two vectors follows the same distribution pattern, however, these are differentiated by a translation on the x axis. The x-axis shift is due to the fact that the initialization passwords that were used only vary by one bit from each other. Each point that moves away from the “line” of linear growth represents the randomness that is introduced in the process of creating the vectors of pseudo-ordered indexes.

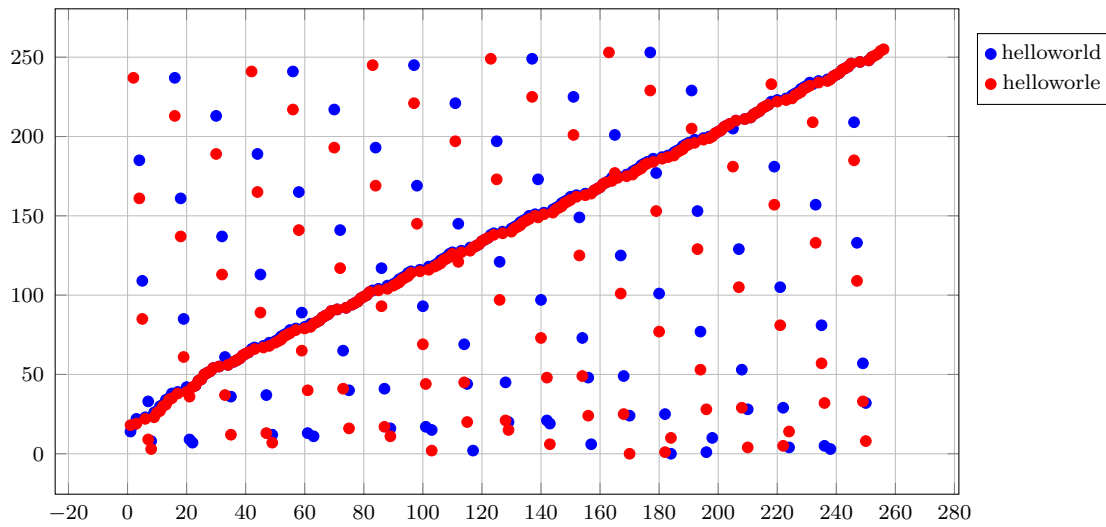


Figure 7.12: Pseudo-order index vector distribution with passwords “helloworld” and “helloworle”

Distribution of Public and Private Key

Figure 7.13 shows the difference between the public and private key created with the password “helloworld”. The values that make up the public key and the private key do not coincide at any point. The private key is “bigger” than the public key in terms of possible values that each key could make up. It can be seen in Figure 7.13 that the private key is made up of values within a range of $[-2.2, 1.5]$ while the public key is made up of values within an approximate range of $[-0.65, 0.65]$.

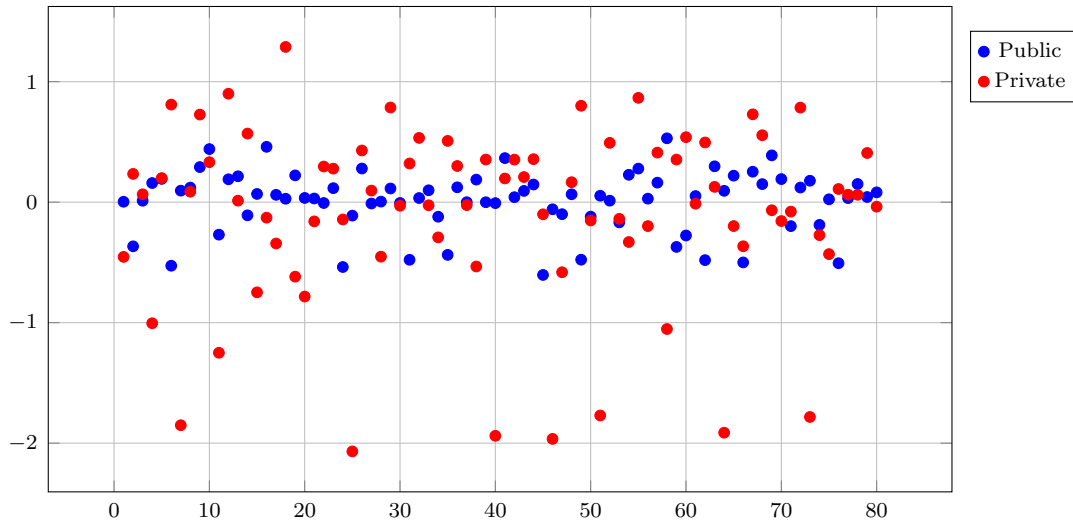


Figure 7.13: Public and Private key distribution with password “helloworld”

Figure 7.14 shows the difference between the keys created with the password “helloworle”. The distribution of the public and private keys has a high randomness and they do not coincide at any point with each other. In Figure 7.14 it can be seen that the private key is larger than the public key. The private key is within a range of $[-2.5, 1.75]$ while the public key is within an approximate range of $[-0.60, 0.60]$.

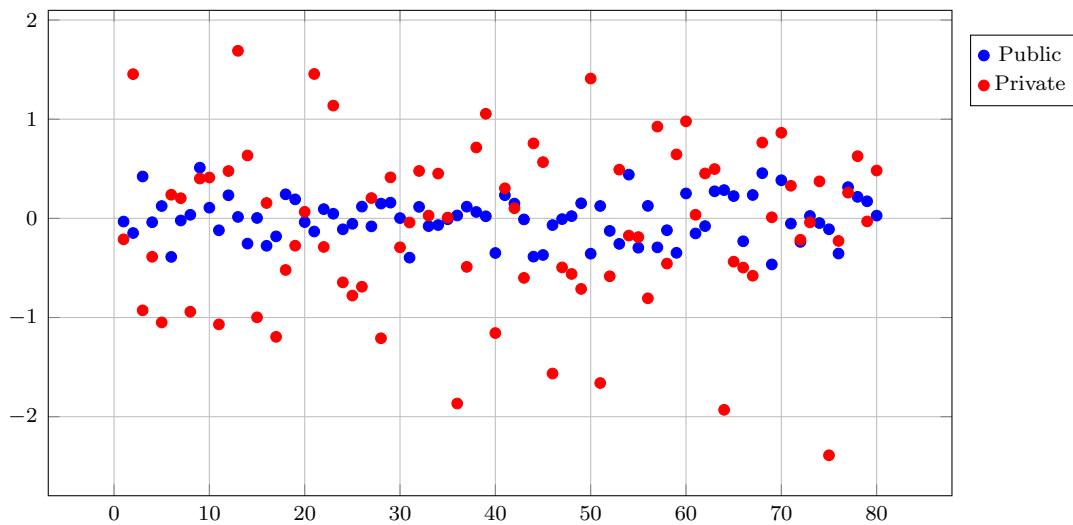


Figure 7.14: Public and Private key distribution with password “helloworle”

The difference between the public and private key is due to the cryptosystem seeks to guarantee a high level of security. Since the public key is publicly known, the possible values that make it up are irrelevant for security. On the other hand, the private key, since it is a secret, needs a method that makes it difficult to replicate it. So the cryptosystem generates private keys with a greater range of possible values.

Public Key Comparison

Figure 7.15 shows the difference between the public key generated with the password “helloworld” and the public key generated with “helloworle”. The distributions of the public keys are very different from each other, even though the initialization password is only varied by one bit. In Figure 7.15 it can be seen that the key corresponding to “helloworld” is in a range of $[-0.61, 0.52]$ while that corresponding to “helloworle” is in a range of $[-0.45, 0.45]$.

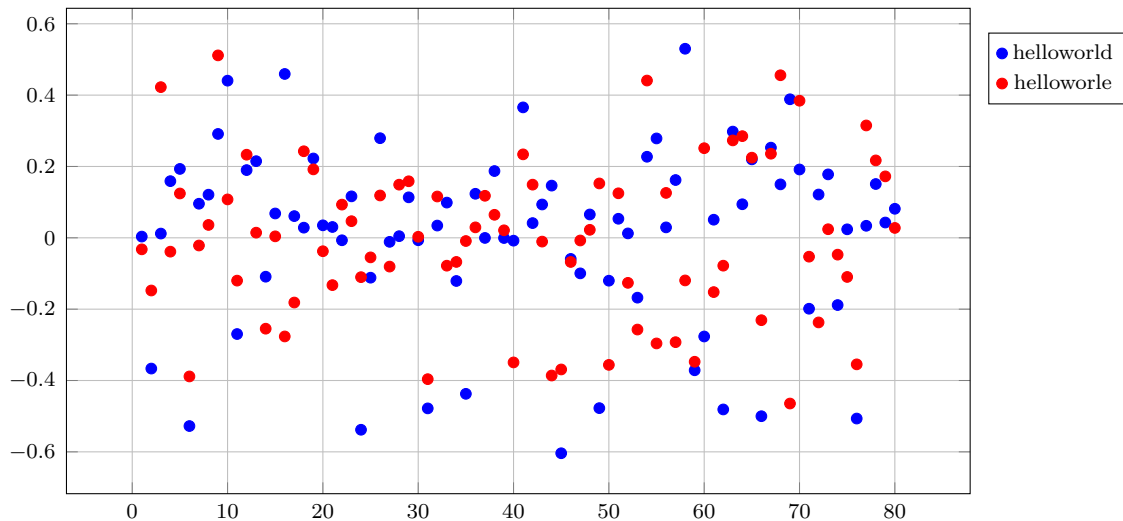


Figure 7.15: Comparison between public keys generated by passwords “helloworld” and “helloworle”

Private Key Comparison

Figure 7.16 shows the difference between the private key generated with “helloworld” and the private key generated with “helloworle”. The private key distributions are very different from each other. In figure 7.16 it can be seen that the key corresponding to ”helloworld” is in a range of $[-2.2, 1.32]$ while the one corresponding to ”helloworle” is in a range of $[-2.5, 1.8]$.

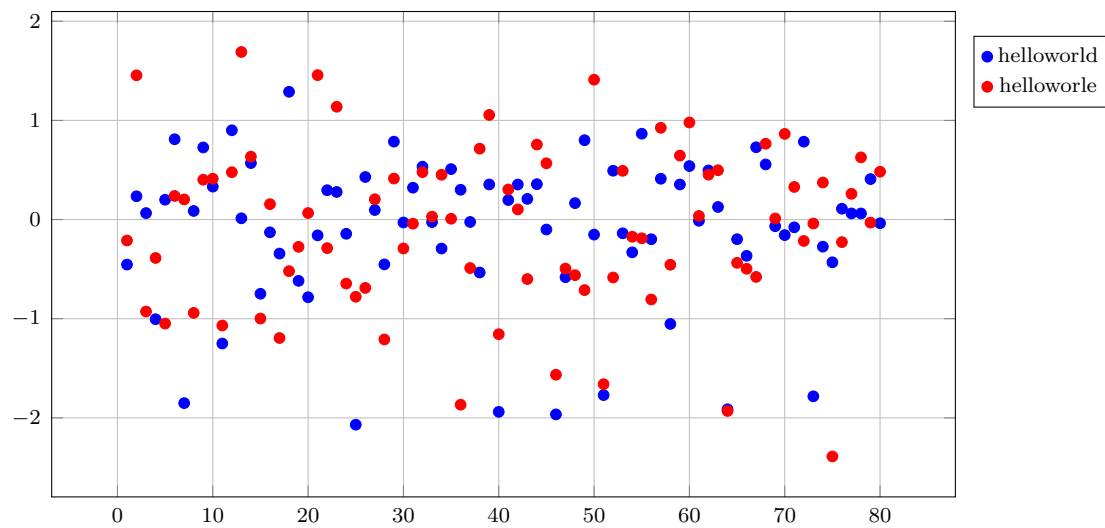


Figure 7.16: Comparison between private keys generated by passwords “helloworld” and “helloworle”

The level of randomness that the public and private keys present when compared to each other indicates that the randomization algorithm that was created together with the cryptosystem works correctly. Additionally, the variation that exists between the ranges of values that make up the keys makes the system have a high level of security.

Chapter 8

Conclusions

Based on the results obtained in this work, we can conclude that the proposed asymmetric key cryptographic system presents a considerable performance improvement with respect to traditional algorithms. In more detail, the following conclusions are pointed out:

1. The experimental results showed that the proposed cryptosystem presents much lower calculation times than other asymmetric algorithms. The file sizes that were used for the purchase varied for each algorithm and ranged from 32 KB to 280 KB. Additionally, the system was evaluated with files with sizes from 200 KB to 600 KB, where the system did not present a major problem in the encryption and decryption of the information. Regarding the key generation time, the system has spends longer execution time than traditional algorithms because it is equal to the time required for the neural network to be trained.
2. The autoencoder neural network architecture that was chosen has a high capacity to encode all the ASCII code in a reasonable computational time. The average training time obtained was 27.47 seconds, which is acceptable due to this process is only performed once per set of keys.
3. The system was able to generate totally different sets of keys with the variation of one bit in the initialization password. Therefore, the randomization algorithm developed for the neural network training process to have a high level of randomness presented successful results.
4. The security evaluation carried out on the cryptographic system determined that a brute force hack, whether trying to “reconstruct” the private key or to compromise the entire system, would require an enormous computational time, making hacking infeasible.

8.1 Future Work

Since the proposed cryptographic system is based on neural networks, as future work, we will implement the cryptosystem under a parallelization approach, thus improving training times (generation of keys). With this approach, the level of randomness could be increased without compromising the performance of the neural network.

Bibliography

- [1] K. Witeczyńska, “Effective protection of information systems and networks against attacks in the era of globalization,” *Logistics and Transport*, vol. 41, 2019.
- [2] V. Pachghare, *Cryptography and information security*. PHI Learning Pvt. Ltd., 2019.
- [3] F. Maqsood, M. Ahmed, M. M. Ali, and M. A. Shah, “Cryptography: A comparative analysis for modern techniques,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017. [Online]. Available: <http://dx.doi.org/10.14569/IJACSA.2017.080659>
- [4] H. Mathur and Z. Alam, “Analysis in symmetric and asymmetric cryptology algorithm,” *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 4, no. 1, 2015.
- [5] C. Narasimham and J. Pradhan, “Evaluation of performance characteristics of cryptosystem using text files.” *Journal of Theoretical & Applied Information Technology*, vol. 4, no. 1, 2008.
- [6] L. Zhinin-Vera, O. Chang, R. Valencia-Ramos, R. Velastegui, G. E. Pilliza, and F. Quinga-Socasi, “Q-credit card fraud detector for imbalanced classification using reinforcement learning.” pp. 279–286, 2020.
- [7] S. Bhowmik and S. Acharyya, “Image cryptography: The genetic algorithm approach,” vol. 2, pp. 223–227, 2011.
- [8] M. Coutinho, R. de Oliveira Albuquerque, F. Borges, L. J. Garcia Villalba, and T.-H. Kim, “Learning perfectly secure cryptography to protect communications with adversarial neural cryptography,” *Sensors*, vol. 18, no. 5, p. 1306, 2018.
- [9] P. Singh and P. Shende, “Symmetric key cryptography: current trends,” *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 12, pp. 410–415, 2014.
- [10] A. Oad, H. Yadav, A. Jain *et al.*, “A review: Image encryption techniques and its terminologies,” *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, pp. 2249–8958, 2014.
- [11] W. Stallings, *Cryptography and network security, 4/E*. Pearson Education India, 2006.

- [12] E. S. Nigm, E.-S. El-Rabaie, O. Faragallah, and A. Mousa, "Cryptography and database security: Concepts, compliance risks and technical challenges," 07 2010.
- [13] S. Tayal, N. Gupta, P. Gupta, D. Goyal, and M. Goyal, "A review paper on network security and cryptography," *Advances in Computational Sciences and Technology*, vol. 10, no. 5, pp. 763–770, 2017.
- [14] Z. Zaru and M. Khan, "General summary of cryptography," *Journal of Engineering Research and Application*, 2018.
- [15] S. Mewada, P. Sharma, and S. Gautam, "Classification of efficient symmetric key cryptography algorithms," *International Journal of Computer Science and Information Security*, vol. 14, no. 2, p. 105, 2016.
- [16] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," in *2017 international conference on engineering and technology (ICET)*. IEEE, 2017, pp. 1–7.
- [17] X. Zhou and X. Tang, "Research and implementation of rsa algorithm for encryption and decryption," in *Proceedings of 2011 6th international forum on strategic technology*, vol. 2. IEEE, 2011, pp. 1118–1121.
- [18] N. Sklavos, "Book review: Stallings, w. cryptography and network security: Principles and practice: Upper saddle river, nj: Prentice hall, 2013, 752p., \$142.40. isbn: 13: 978-0133354690." 2014.
- [19] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [20] K. Huang and R. Tso, "A commutative encryption scheme based on elgamal encryption," in *2012 International Conference on Information Security and Intelligent Control*. IEEE, 2012, pp. 156–159.
- [21] X. Fang and Y. Wu, "Investigation into the elliptic curve cryptography," in *2017 3rd International Conference on Information Management (ICIM)*. IEEE, 2017, pp. 412–415.
- [22] N. Tirthani and R. Ganesan, "Data security in cloud architecture based on diffie hellman and elliptical curve cryptography." *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 49, 2014.
- [23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.
- [24] C. Jost, H. Lam, A. Maximov, and B. J. Smeets, "Encryption performance improvements of the paillier cryptosystem." *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 864, 2015.

- [25] M. S. Hossain, Z. C. Ong, Z. Ismail, S. Noroozi, and S. Y. Khoo, “Artificial neural networks for vibration based inverse parametric identifications: A review,” *Applied Soft Computing*, vol. 52, pp. 203–219, 2017.
- [26] W. J. Park and J.-B. Park, “History and application of artificial neural networks in dentistry,” *European journal of dentistry*, vol. 12, no. 4, p. 594, 2018.
- [27] D. Graupe, *Principles of artificial neural networks*. World Scientific, 2013, vol. 7.
- [28] A. Hajian and P. Styles, “Artificial neural networks,” in *Application of Soft Computing and Intelligent Methods in Geophysics*. Springer, 2018, pp. 3–69.
- [29] S. Sharma, “Activation functions in neural networks,” *Towards Data Science*, vol. 6, 2017.
- [30] H. Li, Z. Zhang, and Z. Liu, “Application of artificial neural networks for catalysis: a review,” *Catalysts*, vol. 7, no. 10, p. 306, 2017.
- [31] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, “An efficient hybrid multilayer perceptron neural network with grasshopper optimization,” *Soft Computing*, vol. 23, no. 17, pp. 7941–7958, 2019.
- [32] V. Pellakuri, D. R. Rao, and J. Murthy, “Modeling of supervised adaline neural network learning technique,” in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2016, pp. 17–22.
- [33] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, “A sparse auto-encoder-based deep neural network approach for induction motor faults classification,” *Measurement*, vol. 89, pp. 171–178, 2016.
- [34] G. Alain and Y. Bengio, “What regularized auto-encoders learn from the data-generating distribution,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.
- [35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [36] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, “Autoencoder-based feature learning for cyber security applications,” pp. 3854–3861, 2017.
- [37] C. Gallo, “Artificial neural networks tutorial,” in *Encyclopedia of Information Science and Technology, Third Edition*. IGI Global, 2015, pp. 6369–6378.
- [38] B. M. Wilamowski, “Neural network architectures and learning algorithms,” *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 56–63, 2009.
- [39] T. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions. McGraw-Hill, 1997. [Online]. Available: <https://books.google.com.ec/books?id=EoYBngEACAAJ>
- [40] J. Li, J.-h. Cheng, J.-y. Shi, and F. Huang, “Brief introduction of back propagation (bp) neural network algorithm and its improvement,” in *Advances in computer science and information engineering*. Springer, 2012, pp. 553–558.

- [41] N. N. Charniya, “Design of near-optimal classifier using multi-layer perceptron neural networks for intelligent sensors,” *International Journal of Modeling and Optimization*, 2013.
- [42] W. Kinzel and I. Kanter, “Interacting neural networks and cryptography,” in *Advances in solid state physics*. Springer, 2002, pp. 383–391.
- [43] N. Sklavos, “Book review: Stallings, w. cryptography and network security: Principles and practice,” *Information Security Journal: A Global Perspective*, vol. 23, no. 1-2, pp. 49–50, 2014. [Online]. Available: <https://doi.org/10.1080/19393555.2014.900834>
- [44] F. Quinga-Socasi, R. Velastegui, L. Zhinin-Vera, R. Valencia-Ramos, F. Ortega-Zamorano, and O. Chang, “Digital cryptography implementation using neurocomputational model with autoencoder architecture,” in *Proceedings of the 12th International Conference on Agents and Artificial Intelligence, ICAART 2020, Volume 2, Valletta, Malta, February 22-24, 2020*, A. P. Rocha, L. Steels, and H. J. van den Herik, Eds. SCITEPRESS, 2020, pp. 865–872. [Online]. Available: <https://doi.org/10.5220/0009154908650872>
- [45] F. Quinga and O. Chang, “A Deep Learning Approach for a Symmetric Key Cryptography System.” Ph.D. dissertation, School of Mathematical and Computational Sciences, Yachay Tech University, March 2020.
- [46] E. Volna, M. Kotyrba, V. Kocian, and M. Janosek, “Cryptography based on neural network.” in *ECMS*, 2012, pp. 386–391.
- [47] R. M. Jogdand, “Design of an efficient neural key generation,” *International Journal of Artificial Intelligence Applications (IJAIA)*, 2011.
- [48] I. Kanter and W. Kinzel, “The theory of neural networks and cryptography,” in *The Physics of Communication*. World Scientific, 2003, pp. 631–642.
- [49] E. Klein, R. Mislovaty, I. Kanter, A. Ruttor, and W. Kinzel, “Synchronization of neural networks by mutual learning and its application to cryptography,” *Advances in Neural Information Processing Systems*, 2005.
- [50] P. P. Hadke and S. G. Kale, “Use of neural networks in cryptography: A review,” in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. IEEE, 2016, pp. 1–4.
- [51] K. Rabah, “Steganography-the art of hiding data,” *Information Technology Journal*, 03 2004.
- [52] J. Davidson, C. Bergman, and E. Bartlett, “An artificial neural network for wavelet steganalysis,” in *Mathematical Methods in Pattern and Image Analysis*, vol. 5916. International Society for Optics and Photonics, 2005, p. 59160D.

- [53] Y. Q. Shi, G. Xuan, D. Zou, J. Gao, C. Yang, Z. Zhang, P. Chai, W. Chen, and C. Chen, "Image steganalysis based on moments of characteristic functions using wavelet decomposition, prediction-error image, and neural network," in *2005 IEEE International Conference on Multimedia and Expo*. IEEE, 2005, pp. 4–pp.
- [54] L. M. Cheng and C. Chan, "Pseudo random generator based on clipped hopfield neural network," in *Proc. of the IEEE International Symposium on Circuit and system*, 1998.
- [55] D. Karras and V. Zorkadis, "Improving pseudo random bit sequence generation and evaluation for secure internet communication using neural networks techquines," *International Joint Conf. on neural networks (IJCNN 2003)*, 2003.
- [56] R. Matta and A. Kumar, "Performance comparison for rsa and ecc in video steganography," *International Journal of Engineering Science and Computing*, p. 4638–4644, 2016.
- [57] S. Farah, M. Y. Javed, A. Shamim, and T. Nawaz, "An experimental study on performance evaluation of asymmetric encryption algorithms," 2012.