



UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

A Deep Learning Approach for a Symmetric Key Cryptography System

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

Autor:

Quinga Socasi Francisco

Tutor:

Ph.D - Chang Tortolero Oscar

Urququí, marzo 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00013-AD

A los 18 días del mes de marzo de 2020, a las 10:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Miembro No Tutor	Dr. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

El(la) señor(ita) estudiante **QUINGA SOCASI, FRANCISCO DE JESUS**, con cédula de identidad No. **1725912065**, 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: **A DEEP LEARNING APPROACH FOR SYMMETRIC KEY CRYPTOGRAPHY SYSTEM (HPC)**, 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.
--------------	--

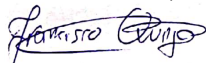
Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

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. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.	10,0
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.	9,5

Lo que da un promedio de: **9.8 (Nueve punto Ocho)**, 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.



QUINGA SOCASI, FRANCISCO DE JESUS
Estudiante

Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
Presidente Tribunal de Defensa

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

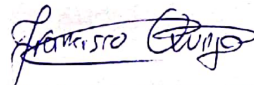
Dr. PELUFFO ORDOÑEZ, DIEGO HERNAN , Ph.D.
Miembro No Tutor

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Francisco de Jesús Quinga Socasi**, con cédula de identidad 1725912065, 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.

Urcuquí, Marzo 2020.



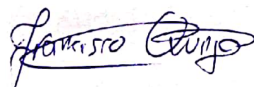
Francisco de Jesús Quinga Socasi
CI: 1725912065

Autorización de publicación

Yo, **Francisco de Jesús Quinga Socasi**, con cédula de identidad 1725912065, 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.

Urcuquí, Diciembre 2019.



Francisco de Jesús Quinga Socasi
CI: 1725912065

Dedication

“This thesis is dedicated to my family how encourage me to pursue my dreams and always support me in all things great and small.”

Francisco de Jesús Quinga Socasi

Acknowledgements

“I would like to take this opportunity to express my thanks to those who helped me with research and writing stuff during the elaboration of this thesis. I would like to express my gratitude to Ph.D. Oscar Chang for his guidance and support throughout the course of this research. I would also like to thank my committee members Ph.D. Freddy Cuenca and Ph.D. Diego Peluffo for their help and support on the writing of this thesis.”

Francisco de Jesús Quinga Socasi

Resumen

En el mundo digital actual, inmerso en toneladas de gigabytes provenientes de transacciones en línea fluyendo constantemente a través de Internet, la seguridad de la información se ha convertido en una necesidad y tema de gran interés. En este sentido, se alienta a los científicos informáticos a proteger la información contra las personas maliciosas, que evolucionan rápidamente con la tecnología. El objetivo principal de la Criptografía es aplicar matemática y lógica complejas para desarrollar procesos de cifrado y descifrado, mediante los cuales se hace que la información sea ininteligible, y se recuperan los datos originales, respectivamente. Evitando así el acceso no autorizado a la información. Hasta la fecha, se han desarrollado varios algoritmos de criptografía. Como AES, 3DES y RSA, donde cada cifrado tiene sus propios puntos fuertes y débiles. Por una parte, los algoritmos de criptografía tradicionales aplican operaciones en serie exhaustivas utilizando fórmulas complejas y números primos enormes, lo que hace que el cifrado y el descifrado sean muy pesados computacionalmente y de alguna manera vulnerables. Por otro lado, recientemente el aprendizaje automático y la inteligencia artificial han logrado mejoras significativas en Criptografía. Este trabajo propone un sistema alternativo de criptografía simétrica basado en aprendizaje profundo utilizando dos componentes principales: (1) Un tipo particular de red neuronal denominada *autoencoder* para el cifrado y descifrado de datos, (2) Un algoritmo de generación de claves que transforma una contraseña alfanumérica en una secuencia de números enteros usados en los procesos aleatorios de la fase de entrenamiento. Los resultados experimentales muestran que el sistema propuesto supera AES, DES, 3DES y RSA al cifrar archivos de tamaño no más de 868 KB. También mostramos que el sistema propuesto puede representar una contribución significativa al campo de la seguridad de datos.

Palabras clave: Aprendizaje profundo, *autoencoder*, cifrado simétrico, criptografía, generación de claves.

Abstract

In today's digital world, immersed into tons of gigabytes coming from online transactions constantly flowing over the Internet, information security has become a necessity and subject of great interest. In this sense, computer scientists are highly encouraged to protect information against malicious people, who evolves fast with technology. The main purpose of Cryptography is to apply complex mathematics and logic to develop encryption and decryption processes, whereby the information is made unintelligible, and the original data is recovered, respectively. Thus avoiding unauthorized access to information. Up to date, various cryptography algorithms have been developed. Such as AES, 3DES, and RSA, where each cipher entails the advantages and drawbacks thereof. On the one hand, traditional cryptography ciphers apply exhaustive serial operations using complex formulas and huge prime numbers, making encryption and decryption computing consuming and somehow vulnerable. On the other hand, recently machine learning and artificial intelligence have achieved significant improvements in Cryptography. This work proposes an alternative deep learning encryption system with two principal components: (1) A particular type of neural network called *autoencoder* for encryption and decryption of data, (2) A key generation algorithm which transforms an alphanumeric password into a sequence of integer numbers used during the random processes of the training phase. Experimental results show that the proposed system overcome AES, DES, 3DES, and RSA when encrypting files of size no longer than 868KB. We also show that the proposed system may represent a meaningful contribution to the field of data security.

Keywords: *Autoencoder*, cryptography, deep learning, key generation, symmetric encryption.

Contents

1	Introduction	5
1.1	Problem Statement	6
1.2	Justification	6
1.3	Contribution	6
1.4	Thesis overview	7
2	Objectives	8
2.1	General Objective	8
2.2	Specific Objectives	8
3	Theoretical Framework	9
3.1	Cryptography	9
3.1.1	Basic Concepts and Terminology	9
3.1.2	Goals of Cryptography	10
3.1.3	Modes of Ciphers	11
3.1.4	Classification of Cryptography Algorithms	12
3.1.5	Performance of Cryptography Algorithms	16
3.1.6	Advantages and Disadvantages for Symmetric Key Cryptosystems	18
3.1.7	Cryptanalysis	19
3.1.8	Vulnerabilities of Classical Cryptographic Algorithms	20
3.2	Deep Learning	21
3.2.1	Toward Deep Learning	21
3.2.2	Exploring an Artificial Neural Networks	22
3.2.3	Learning Styles	26
3.2.4	The Architecture of Neural Networks	31
4	State of the Art Review in Cryptography with Deep Learning	37
4.0.1	Neural cryptography	37
4.0.2	Pseudo random generators	38
4.0.3	Neural attacks	39
4.0.4	Visual cryptography	39
4.0.5	DNA cryptography and Deep Learning	41

- 5 Methodology 42**
 - 5.1 Research Approach 42
 - 5.2 Encryption and decryption approach 42
 - 5.3 Symmetric Key Cryptography Model 44
 - 5.3.1 Key generator 45
 - 5.3.2 Autoencoder architecture 47
- 6 Experimental Setup 49**
 - 6.1 Software and Hardware 49
 - 6.2 Performance Evaluation Parameters 49
 - 6.2.1 Encryption time 49
 - 6.2.2 Decryption time 50
 - 6.3 Security Analysis Parameters 50
 - 6.3.1 Accuracy 50
 - 6.3.2 Sensibility of the network 50
 - 6.3.3 Cut-off point 50
 - 6.4 Performance Experiments 50
 - 6.5 Security Experiments 51
- 7 Results and Discussion 53**
 - 7.1 Experiments 53
 - 7.1.1 Experiment 1: Measuring the encryption time 53
 - 7.1.2 Experiment 2: Measuring the decryption time 56
 - 7.1.3 Experiment 3: Identifying the cut-off point 58
 - 7.1.4 Experiment 4: Reliability of the key generator 60
 - 7.1.5 Experiment 5: Attacking with brute force approach 60
- 8 Conclusions and Perspectives 61**
- References 64**
- Appendices 72**
- A Cryptosystem Prototype 74**
 - A.1 GUI for monitoring the neural network training 74
 - A.2 Encryption and decryption demonstration 75

List of Figures

3.1	DES structure	13
3.2	AES structure	14
3.3	Model of an artificial neuron	23
3.4	Three layer neural network	25
3.5	Backpropagation learning rule.	26
3.6	Different machine learning techniques	27
3.7	Single-layer feedforward network	32
3.8	Multilayer Feedforward Network.	33
3.9	Recurrent Network.	34
3.10	Mesh Network.	34
3.11	Architecture of an Autoencoder	35
5.1	Encryption Process.	43
5.2	Decryption process.	44
5.3	Overall flowchart of the proposed symmetric key cryptography system.	45
5.4	Seed Sequence Generation Flowchart	46
5.5	Autoencoder Architecture of the Proposed Model	48
7.1	Comparative at Encryption Time with Patil et al.	54
7.2	Comparative at Encryption Time with Rihan et al.	55
7.3	Comparative at Encryption Time with Mahajan & Sachdeva	56
7.4	Comparative at Decryption Time with Patil et al.	57
7.5	Comparative at Decryption Time with Mahajan & Sachdeva.	58
7.6	Sensibility of the network against different noise levels in the domain of the hundredths domain	59
A.1	GUI for monitoring the autoencoder training. (A): W_E and W_D representation, the red signal means excitatory response while the blue one means an inhibitory response, (B) Expected and actual values, the former is represented by red lines and the latter by green lines, (C) Hidden values representation, i.e the cipher code, (D) Reports about the number of retrieved patterns and elapsed epochs.	75
A.2	Depiction of the functionality of the proposed system.	76

List of Tables

3.1	Comparison Between Symmetric and Asymmetric Encryption.	17
3.2	Comparative at the Throughput of an Encryption and Decryption Algorithms with Key and Block Size.	17
3.3	Comparative at Vulnerabilities of Classical Classical Cryptographic Algorithms.	21
3.4	Transfer Functions and their Corresponding Equations for Computation.	24
3.5	Comparison Between Traditional Computing and ANNs.	26
5.1	Hyperparameters List of the Proposed Neural Netwrok	48
7.1	Comparative at Encryption Time with Patil et al.	54
7.2	Comparative at Encryption Time with Rihan et al.	55
7.3	Comparative at Encryption Time with Mahajan & Sachdeva	56
7.4	Comparative at Decryption Time with Patil et al.	57
7.5	Comparative at Decryption Time with Mahajan & Sachdeva.	58
7.6	Network Sensibility Against Different Decimal Units, namely: Thousandths, Hundredths and Tenths.	59
7.7	Average Distance Between Synaptic with Respect to the Password "hello".	60

Chapter 1

Introduction

Cryptography is called the art of secret communications, it is the science of information and communication security [1]. Their techniques have been used since thousands of years ago to disguising a message with the objective that only the intended recipient can read it, avoiding message interception by non-authorized people and falling into wrong hands [2]. Therefore, cryptography has one of the most important and challenging roles of the digital world where data security is omnipresent. Indeed, cryptography is widely used for authentication in bank cards, social networks, e-commerce, and streaming services. Likewise, it is used in access control applications upon car lock systems and ski lifts. Also, it is present in payment methods like prepaid telephone cards and e-cash [1].

Nowadays, there are plenty of algorithms performing the process of encoding and decoding information before sending it over a public communication channel, which is considered as an insecure transmission canal, like the internet or a WLAN [3]. These algorithms are classified according to the type of encryption they perform: in symmetric key, asymmetric key, and hash functions. Among the symmetric key algorithms we have: DES , AES and 3DES [4]. Likewise, the asymmetric key algorithms are: RSA (Rivest, Shamir y Adleman), DSA (Digital Signature Algorithm), and ECC (Elliptic Curve Cryptography) [3]. Between the most common hash functions we have: MD5 and SHA-2 [5]. According to Medina and Miranda (2015) [4], there are two main characteristics to identify and differentiate the algorithms of encryption. The first is their capacity to secure the data against attacks (how difficult it is to crack the algorithm) and second is the velocity and efficiency to perform the encryption and decryption tasks.

In the digital world, a relatively new computational set of tools, have been developed to treat and solve many complex real-world problems, these tools are the artificial neural networks (ANNs). The attractiveness of ANNs is their capability to processing information in a nonlinear way, they are high parallelizable, fault and noise-tolerant, and feature learning capabilities [6]. Due to all of the advantages that ANNs hold, they are used in many fields of science with different approaches.

The purpose of this project is to implement an alternative encryption system, avoiding the use of traditional sequential algorithms. This study takes advantage of the potential that deep learning and ANNs offer. Using a particular type of ANN called autoencoder, we will generate a stable method of encryption and decryption. In this project, the information that will be encrypted is the 256 ASCII characters. All the characters will be encrypted and decrypted by

using the hidden layers of a deep ANN. To verify the validity of the proposed encryption system, we will evaluate the two characteristics that were mentioned previously, namely, how difficult is to break the encryption system and the velocity and efficiency of encoding and decoding the data.

1.1 Problem Statement

In this work, the problem statement lies in the lack of a cipher method for encrypting the full set of ASCII characters. Then, the goal for this degree thesis is to develop a symmetric key cryptography system to effectively encrypt and decrypt the complete set of 256 ASCII characters. This thesis is focused on the development of a more reliable and effective cryptography algorithm as compared with traditional symmetric key ciphers. The problem formulation is two-fold: (1) We find the effective topology and parameters that enable an autoencoder to successfully encrypt and decrypt 256 binary 8-dimensional vectors which correspond to the characters in the extended ASCII table, (2) Train and evaluate the robustness of the proposed system by analyzing how difficult braking the system would be.

1.2 Justification

In its purest form, the strength of commonly used cryptographic algorithms relies on complex mathematics like modular arithmetic, complex prime number calculations, and discrete mathematics. In addition, all of these encryption techniques are based on a serial performance, it means a considerable amount of computations. Although conventional cryptography algorithms require high computational power, they are still susceptible to attacks, which can recover the key and therefore retrieve the original message. Therefore, a deep learning-based approach is applied in this thesis using a type of ANN knows as autoencoder. ANNs give a worthy solution to this problem in the sense that neural networks and cryptography can be merged to increase the security level of the complete system while simplifying the computations.

1.3 Contribution

One of the contributions of this work is to propose a new symmetric key encryption and decryption system for text messages to overcome existing security problems in traditional ciphers by using an autoencoder ANN architecture and a key generation mechanism that increases the security of the system.

Another contribution of this work is to improve the state of the art with respect to the use of neural networks for cryptography. Several researchers have used neural networks to either encrypt images or just as a key interchange protocol. However, the scheme proposed in this work actually encrypts and decrypts text messages.

1.4 Thesis overview

This thesis is organized into 8 main Chapters named as follows: Introduction, Objectives, Theoretical Framework, Methodology, Experimental setup, Result; and Conclusions and Future work

In Chapter 1, the problem statement, the justification of this work, and the scientific contributions of this research are presented.

Chapter 2 states the general objective and a list of specific objectives.

The explanation about cryptography and deep learning is explained in Chapter 3

discussion about previous research about the use of deep learning for cryptographic purposes is carried out in Chapter 4

In Chapter 5 it is discussed the methodology implemented to set up the system and analyze its robustness against attacks.

The results obtained in this thesis are presented in Chapter 7.

Finally, in Chapter 8, the conclusions obtained from this work are presented. Also, future works that can improve the proposed methodology and help to establish open issues are mentioned.

Chapter 2

Objectives

2.1 General Objective

The general objective of the project is to implement an alternative symmetric encryption system by combining artificial neural networks (ANNs) and deep learning techniques to overcome the weaknesses of conventional symmetric ciphers.

2.2 Specific Objectives

The specific objectives of the project include:

- To cipher the entire 256 ASCII characters in binary representation by using an ANN architecture called autoencoder as a cipher algorithm to generate a stable cryptography system.
- To use different seeds in every random process involved in training the neural network by implementing a key generator that produces a private key in the form of a number sequence to increase the entropy level of the proposed system.
- To guarantee data confidentiality by analyzing how difficult breaking down the system will be to show the security level of the proposed cryptography system.

Chapter 3

Theoretical Framework

3.1 Cryptography

At the present days, information has become an essential commodity and the access of it through online resources has become a common routine of our daily basis. However, the information being transmitted through wireless networks always is susceptible to active and passive attacks [7]. Therefore, to enhance data protection, information security is indispensable. This is achieved using cryptography techniques. Cryptography can be defined in several ways. According to [7], cryptography is the art and science of secret writing that changes the original structure of a message into a twisted structure. Similarly, in [8] cryptography is more intuitive defined as computer science and mathematics that enables secure communication between two parties while there is an eavesdropper between them.

3.1.1 Basic Concepts and Terminology

Some of the basic terminology to understand cryptography includes the following terms:

- *Plain Text*: Is the original message that is going to be transmitted over the network. For example, if Alice wants to say “*We shall meet behind the monument in the garden.*” to Bob, then “*We shall meet behind the monument in the garden.*” is the Plain Text [7].
- *Cipher Text*: The original message processed such that it cannot be understood by a man in the middle is called cipher text. For example, after applying Caesar’s cipher key with key = 13 to the message ”“*We shall meet behind the monument in the garden.*” the resulting Cipher Text is “*Jr funyy zrrg oruvaq gur zbahzrag va gur tneqra*” [7].
- *Encryption*: The conversion of Plaint Text into Cipher Text, using a parameter known as a Key and a specific encryption algorithm, is called Encryption [9]. A variety of encryption methods are used by cryptographers when sending confidential information through insecure channels. The sender is responsible for performing the encryption process.
- *Decryption*: The opposite operation of Encryption is known as Decryption. It is the process of retrieving Plain Text from the Cipher Text using a decryption algorithm and a Key [9]. Overall, Encryption and Decryption algorithms are equivalent but reverse [7].

- *Key*: Parameter, usually a string of alphanumeric characters used to encrypt and decrypt the message [7]. The functionality of a cryptography algorithm is given by the Key, i.e without a key the algorithm do noting [9]. In particular, using a Hill Cipher with key [11 10 20 09] to encrypt the Plain Text “*Gold is buried under the bush of Red Roses!*” the Cipher Text “*ymvnikdshwwdmxvsnrnudsihwntmyfwaqi*” is obtained [7]. During the encryption process, the key operates on the Plain Text, while it operates on the Cipher Text during the decryption step
- *Key Management*: The set of procedures for supplying the generation, distribution, tracking, control, storage, protection, and destruction for all cryptographic key material, comprising symmetric keys as well as public keys [9].
- *Cipher*: A specific algorithm used for encryption and decryption purposes is called Cipher. There is two kinds of Ciphers, namely Block Ciphers and Stream Ciphers [9].
- *Encoder*: The entity that wishes to send a message in a secure fashion. It uses an encryption algorithm.
- *Decoder*: This is the entity that decrypts the received unreadable message. It could be the intended receiver or either an intruder.
- *Key Strength*: The strength of the Cipher Text is given by the degree of entropy that the Key and the Cipher can produce. Key selection and attributes are important factors to determine the entropy if the cryptography system. To avoid brute force attack the Key length should be large enough. In addition, similar strengths can be achieved using different Key lengths and Ciphers. For example, a 128-bit Key length under an asymmetric approach is pretty similar in security to an 80-bit Key length in a symmetric algorithm [9]

3.1.2 Goals of Cryptography

Good cryptography is focused on the principles of confidentiality, data integrity, authentication, and non-repudiation.

Confidentially

It can be understood as the set of operations to guarantee that only the authorized person has access to the data [10]. There are several methods to provide confidentiality such as physical protection and encryption algorithms to generate diffuse data representations [11], [12]. Confidentiality establishes a set of rules to limit access to certain data [8].

Data Integrity

It can be defined as the set of operations to guarantee a correct data delivery without modifications on it [10]. Common data modifications include substitution, deletion and insertion [11], [13]. In other words, data integrity ensures the consistency and correctness of the information during its entire lifetime [8].

Authentication

It is the set of operations to ensure the received information comes from the real sender only [14]. Authentication verifies the truth of the data attributes that are claimed to be true by someone [8]. In this sense, a user has to show its identity to someone who is not aware of their identity.

Non-repudiation

It is described as the set of operations to guarantee that both the sender and receiver must not reject the interchange of information, this means that the sender of a message can not reject he/she did not send it [10]. In this way, the inability to compose a piece of information to deny data transmission is ensured.

3.1.3 Modes of Ciphers

Depending on whether the plain text is divided into chunks or kept its original structure before being encrypted the cryptography algorithms are divided into block and stream ciphers.

Block ciphers

Basically speaking, a block cipher is an encryption algorithm that works on fixed-size blocks of data and uses the same key to encrypt and decrypt the message [15]. It processes the entire data separately. In a more mathematically way a block cipher is defined as a cipher acting on Σ^n for a given *block size* $n \in \mathbb{N}$ and *alphabet* Σ [8]. Block cipher is the base for most real-world cryptography applications. For instance, they are used to build stream ciphers and cipher with even strong security properties [16]. Syntactically speaking, a block cipher is just a particular kind of cipher, but its semantic security is far stronger: the permutation of a randomly chosen key also looks random [16]. However, the block size in any practical ciphers is very short. For example, AES is 128-bit long. This forces to break down the message when long data transmission. This mode of operation where the information is partitioned is called *electronic codebook mode*. The most important block ciphers include:

- DES.
- AES.
- Blowfish.
- XTAE (Extended Tiny Encryption Algorithm).
- RC6.
- Serpat.

Stream ciphers

Unlike block cipher, stream cipher can handle information of any size without the requirement of having blocks [8]. In a mathematical context, a stream cipher is defined as a cipher acting on Σ^* for any given size and alphabet Σ where each plain text digit is encrypted one at a time with the corresponding digit of a keystream [8], [15]. The keystream is defined as a stream of random or pseudo random characters or digits from Σ that are combined with a plain text to generate a cipher text digit [8]. Usually, the keystream is combined with the plain text to produce the cipher text by simply using and XOR other function [17]. The basic idea of a stream cipher is that instead of having a message partitioned in fixed-size large blocks, like 128 bits, a stream cipher uses very short blocks and makes the encryption dependent on the previous blocks. In addition, depending on the nature of the keystream a stream cipher can be *synchronous* or *self-synchronizing* [18].

- *Synchronous stream cipher*: It is a stream cipher where the keystream is independent from plain text and cipher text. It only depends on the previous ones and the initial key.
- *Self-synchronizing stream cipher*: In contrast with a synchronous stream cipher, a self-synchronizing stream cipher is a stream cipher where the key depends only on the plain text and cipher text.

3.1.4 Classification of Cryptography Algorithms

Once encrypted a message, we can use either the same key or a different one to decrypt it. These two types of cryptographic approaches are known as symmetric encryption and asymmetric encryption, respectively.

Symmetric Cryptography

Also known as single-key cryptography, since it has only one key. In this encryption process, the receiver and the user have to agree on a single secret key which is shared. When a message is encrypted, unintelligible data is produced, with approximately the same length as the original text. In order to decrypt the message, the inverse of the coding and the same encryption key are used. Among the well known symmetric encryption algorithms are: *DES*, *AES* and *3DES* [4].

1. **DES (Data Encryption Standard)**: DES was developed by IBM in the 1970s but was later adopted by the National Institute of Standards and Technology (NIST). It is based on an encryption algorithm known as Feistel block cipher [7]. The Data Encryption Standard (DES) is a Block Cipher which is designed to encrypt and decrypt blocks of data consisting of 64 bits by using a 64-bit key [19]. In the beginning, a permuted choice selects 56 bits from the initial 64. The remaining bits are used as a parity check. The 56 bits are divided into two 28-bit parts. In the subsequent rounds, each part is rotated left one or two bits and a permuted choice produces the final 48-bits result. Figure 3.1 depicts the DES cipher structure

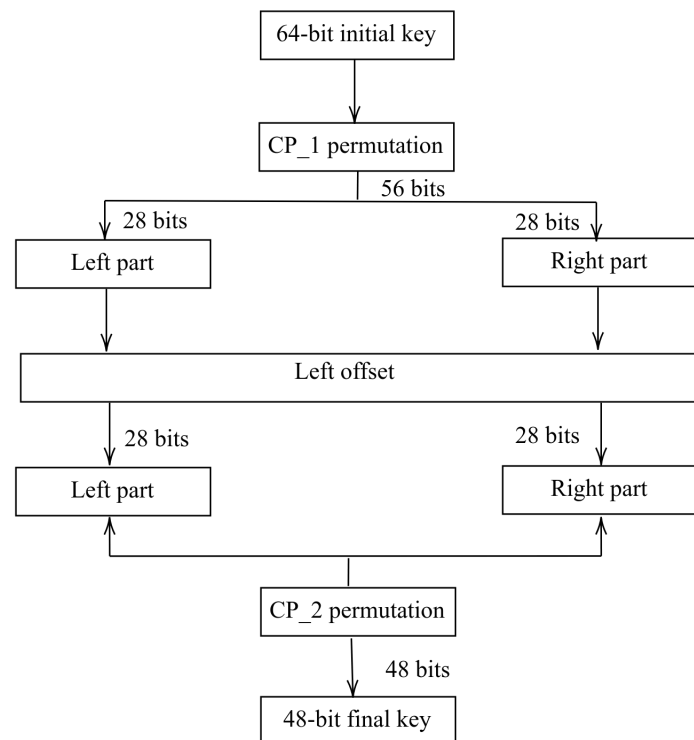


Figure 3.1: DES structure

2. **3DES (Triple Data Encryption)** 3DES or the Triple Data Encryption Algorithm (TDEA) was developed to solve the security weaknesses in DES. Because DES uses a 56-bit key and is not enough to encrypt secret data. 3DES extend the key size of DES by applying the algorithm three times in succession with three different keys to each block of the message. Therefore the key size is thus 168 bits (3 times 56) [19].

3. **AES (Advanced Encryption Standard):** AES is the new encryption standard, also based in blocks, recommended by NIST to replace DES in 2001. AES algorithm can encrypt any data string of 128 bits with keys length of 128, 192, and 256 bits. The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length. The blocks in AES are treated as an array of bytes and organized as a matrix to perform the cipher [19]. see the figure 3.2

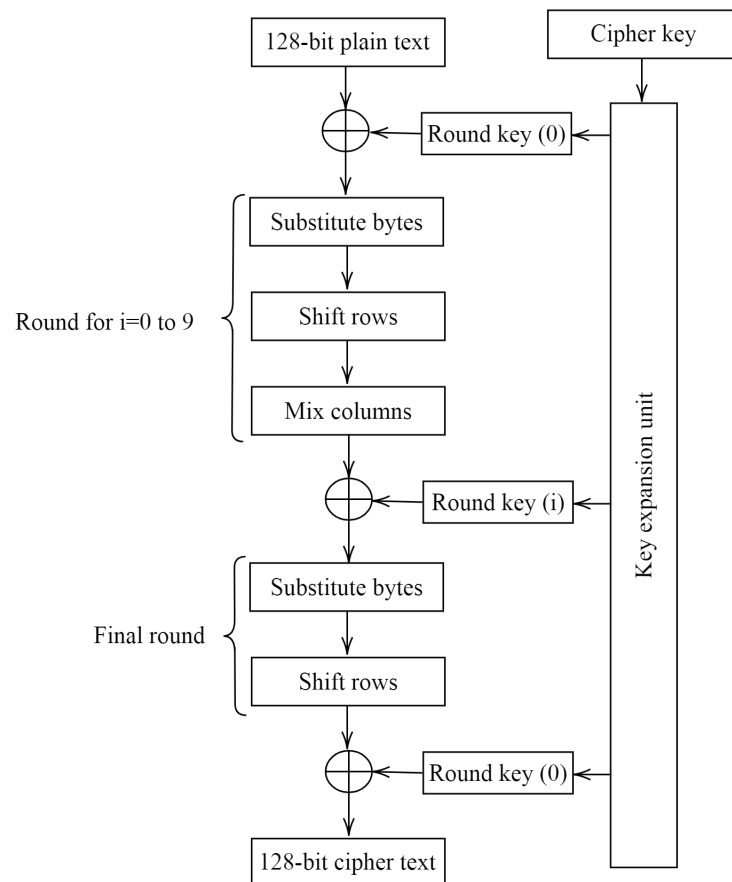


Figure 3.2: AES structure

Asymmetric Cryptography

Asymmetric (or public-key) cryptography is a technology that deals with communication over open networks. The cryptography of public key, uses as a base of encryption, the difficulty of solving certain mathematical problems and functions of a single way [3]. The main feature of public key cryptosystems is that the encryption and decryption procedures are performed with two different keys: public key and private key. One of the most important facts to consider the asymmetric encryption secure is that the private key can not be derived from the public key. Therefore, do not exist the risk of leaking the secret information [20]. Popular asymmetric techniques includes: *RSA*, *DSA* and *ECC*

1. **RSA (Rivest, Shamir y Adleman):** RSA is the first algorithm that can be used both for data encryption and digital signatures. In the algorithm, two large prime numbers are used for constructing the public key and the private key. It is estimated that the difficulty

of guessing the plain text from the signal key and the cipher text equals to that decomposition of the product of two large prime numbers. RSA is mostly used for, encryption of small amounts of data and digital signatures [20].

Given,

P = plain text

C = cypher text

(n, e) = public key

(n, d) = public key

Encryption:

$$C = E(P) \equiv P^e \pmod{n} \quad (3.1)$$

Decryption:

$$P = D(C) \equiv C^d \pmod{n} \quad (3.2)$$

2. **DSA (Digital Signature Algorithm):** DSA is a method for generating and verifying a digital signature of a message m . According to Kravitz and Mills [21], this method requires a pair of corresponding public and secret keys (y and x), as well as a pair of public and secret values (r and k) for each message. Then the message m , with the signature (r, s) is then transmitted and the received values of r and s are tested to determine whether they are congruent to $0 \pmod{g}$.

To calculate:

$$r = (g^k \pmod{p}) \pmod{q} \quad (3.3)$$

and

$$s = k^{-1}(H(m) + xr) \pmod{q}, \quad (3.4)$$

where H is a known conventional hashing function.

3. **ECC (Elliptic Curve Cryptography):** ECC offers considerably greater security for a given key size. Consequently, a key with smaller size makes it possible a much more compact implementations for a given level of security which means faster cryptographic operations, running on smaller chips or more compact software. In general, the elliptic curve cryptography is a public key crypto-system for mobile/wireless environments.(ECC). The security due to ECC relies on the difficulty of Elliptic Curve Discrete Logarithm Problem [22].

Let P and Q be two points on an elliptic curve such that,

$$kP = Q; \text{ s.t. } k \in \mathbb{R} \quad (3.5)$$

Given P and Q , it is computationally infeasible to obtain k . If k is sufficiently large, k is the discrete logarithm of Q to the base P .

The operation involved in ECC is related to the point multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve.

Hash Algorithms for Encryption

Hash encryption techniques are based on hash functions, which uses a mathematical transformation to encrypt information irreversibly [10], [23]. Hash functions were introduced in the decade of 1970 as a powerful tool to solve several security problems in telecommunications and computer networks [23]. It is a function that maps messages of varying size to fixed-size data, usually between 128 and 512 bits [5]. Hash functions are mainly applied in hash tables. A hash table is a data structure that makes it easy to organize data in such a way it is easy to find them later. Every position in a hash table is known as a *slot* and it stores a data item. An index integer value is assigned to each slot. Every slot is referenced by a key and the data item stored in it is called a value [23]. In hash function cryptography do not exist the traditional concept of key since the cipher text can not transform into plain text [10].

Different properties of a hash function are always expected. For example, a hash function H is one-way. It means that given a randomly chosen output y , it is hard or impossible to find a message x satisfying $H(x) = y$. Another important fact is that the hash of a message acts as a *fingerprint* since different messages have different hashes. In addition, it is clear that the output space of a hash function is larger than the input space. This could cause *collisions*. However, in practice, finding two messages with the same hash value is not feasible [5].

During several years, designing a cryptographic hash function was considered simple. In this sense, two common hash functions called MD5 and SHA-1 were widely used with the hope to be safe. However, weaknesses on MD5 were found but SHA-1, which is based on the same principles as MD5, was believed to be secure and therefore was used for a long time. Nevertheless, with the technological revolution, attacks on both MD5 and SHA-1 finally emerged. The attacks show that breaking down these systems is easier than expected. Therefore, nowadays it is not recommendable to use MD5 nor SHA-1. Alternatives options are available but those are also based on the underlying principles of MD5 and SHA-1. In addition, changing from one hash function to another has considerable software and hardware implications. Thus, evidence of safeness is required to effectively switch. Apart from that, to increase data security NIST has started an open competition to define a new set of hash functions, which will be called SHA-3 and are intended to overcome drawbacks in SHA-2 [5].

3.1.5 Performance of Cryptography Algorithms

A comparative study in selected symmetric and asymmetric techniques was carried out by Tripathi and Agrawal [24]. In that study, the encryption ratio is cataloged as minimum, moderate, and maximum. The speed is defined as fast, slow, and moderate. Tenability is specified as yes or no. The key value is evaluated with regard to the bit value used. Throughput and power consumption are determined as high and less. The experimental results are presented in Table 3.1.

A comparison of the *throughput* between symmetric key schemes was conducted by Nigm, Faragallah & Mousa [9]. This measure indicates the encryption speed and it is obtained by computing the total plain text in bytes encrypted divided by the encryption time [9]. The results of the comparative analysis are shown in Table 3.2

Table 3.1: Comparison Between Symmetric and Asymmetric Encryption.

Parameter	Symmetric encryption			Asymmetric encryption
	DES	3DES	AES	RSA
KEY USED	Same key used for encryption and decryption	Same key used for encryption and decryption	Same key used for encryption and decryption	Different key used for encryption and decryption
THROUGHPUT	Lower than AES	Lower than DES	High	Low
ENCRYPTION RATIO	High	Moderate	High	High
TUNABILITY	No	No	No	Yes
POWER CONSUMPTION	Higher than AES	Higher than DES	High	High
KEY LENGTH	56 bits	112 to 168 bits	128, 192, or 256 bits	>1024 bits
SPEED	Fast	Fast	Fast	Fast

Table 3.2: Comparative at the Throughput of an Encryption and Decryption Algorithms with Key and Block Size.

	DES	3DES	AES
Key size	64	192	256
Block size	64	64	128
Encryption Throughput (Kbytes/s)	4.14	3.54	4.28
Decryption Throughput (Kbytes/s)	6.50	5.80	6.61

3.1.6 Advantages and Disadvantages for Symmetric Key Cryptosystems

Symmetric key cryptography has a couple of advantages and disadvantages [25], [14].

Advantages

- Symmetric key algorithms can achieve high data throughput rates. Certain hardware implementations can encrypt hundreds of megabytes per second. Likewise, some software implementations are able to encrypt a considerable amount of megabytes per second [10].
- One of the deficiencies of asymmetric methods is that they require complex and long computations. Whereas, symmetric-key techniques are relatively fast. As an example, several Solid State Drivers (SSD), which are famous for being extremely fast, uses symmetric key mechanisms to store the data [25].
- Symmetric key methodologies uses relatively short key sizes.
- In symmetric key approaches the encrypted information can be transferred through insecure channels. Since there is no public key, in case that the encrypted information is intercepted, the probability to decrypt it is null. [5]. Only the owner of the secret key is able to decrypt the information.
- To check receiver identity, symmetric key algorithms uses password authentication [25].

Disadvantages

- To ensure confidentiality, the key must keep secret between the parties involved in a communication [10]. In this way, there are difficulties when sharing the key with the intended receiver. Symmetric key is more useful when encrypting your own information than when sharing encrypted information.
- Large networks use many pairs of key and they need a proper administration. In particular, key administration is carried out by an unconditionally trusted Trusted Third Party (TTP) [10].
- When an intruder gets the secret key, the can decrypt the entire set of messages encrypted with that key. In the case of two-way communications, both sides of the data are vulnerable [25].
- Every provided signature should not be repudiated [25].
- To step up security, keys must be renewed frequently. Sometimes it is advisable to use a different key in each conversation [5].

3.1.7 Cryptanalysis

So far, it is clear that cryptography is the process to hide original data representation by using an encryption mechanism. On the contrary, cryptanalysis works in the opposite way. It is a method used for attackers to break down and analyze cipher text to retrieve the plain text without knowing the key [26]. Since there is no standard textbook, the study of cryptanalysis is a mathematical challenge [9]. In this context, two are the main purposes of an attacker. First, they got the cipher text and look for the key to obtain the plain text. Second, they got the cipher text and try to find a mechanism to get the plain text without the key [11]. Cryptanalysis involves a special set of tools like pattern finding, mathematics, and analytical reasoning [10]. In addition, cryptanalysis methods can be divided into *active* and *passive* attacks [27].

In an active attack, the attacker tries to actively change the structure, for example altering or creating new blocks, of the data and resend it to the recipient to pretend it is the original sender [8]. On the contrary, in a passive attack, the eavesdropper only reads the message without any modification on it. In addition, there are other types of attacks based on the key or hash function.

Attacks on key based cryptography

- *Cipher text only attack*: The eavesdropper only obtains a sample of the cipher text without its associated plain text.
- *Known plain text attack*: The attacker gets both the cipher and plain text.
- *Chosen plain text*: The attacker is able to decide how long the plain text should be to get the cipher text.
- *Adaptive Chosen plain text attack*: An attacker uses a special hardware for this attack but it is not possible to extract the encryption key.
- *Brute force attack*: The minimum number of computations is given by the key size
- *Related key attack*:: In this attack, the attacker observes the effect of the cipher text under different keys. It provides a mathematical relation between the keys.
- *Differential attack*: Here the attacker tries to identify a non random behavior on the cipher text and exploits it to recover the key.

Hash functions attacks

- *Collision attacks*: It tries to identify a pair of messages that has the same hash value
- *Pre-image attack*: It is more specific in the sense that identify a message that has a specific hash value.
- *Birthday attack*: This attack exploits the higher probability of collisions between attack attempts and a fixed number of permutations

- *Rainbow table*: Is a cracking password mechanism that uses a predefined table for inverting a hash function.
- *Distinguishing attack*: Here the eavesdropper is capable to recognize between encrypted and random data by analyzing encrypted data samples.
- *Side channel attack*: It takes advantage of hardware implementations of cryptosystem.
- *Dictionary attack*: This attack searches likely possibilities to break a cipher.

3.1.8 Vulnerabilities of Classical Cryptographic Algorithms

DES

DES algorithm suffers from Simple Relations in their keys, i.e. in DES there is a direct relationship between key and resulting cipher text. This fact reduces its strength by one bit [28]. Likewise, DES can be broken by using a linear attack. In this approach, the algorithm can be broken using 2^{43} plain texts [28]. In general, DES is vulnerable to key attacks when a weak key is used [29]. Also, the number of rounds is exponentially proportional to the amount of time required to find a key by a brute force approach [24]. Another weakness of DES was explored by Alani [30], who proposed a known plain text attack based on neural networks. In this attack, an average of 2^{11} plaintext-ciphertext pairs and 51 minutes are required.

3DES

Since triple DES is the same as DES but with three times more operations, it was subject to the same cryptanalytic attacks than DES. Furthermore, 3DES can be broken by known plain text attacks. For example, the attack introduced by Lucks [31] requires approximately 2^{32} known plain texts, 2^{113} steps, 2^{90} single DES encryptions, and 2^{88} memory. Likewise, the attack proposed by Alani [30] requires only 2^{12} plain/cipher text pairs and takes 72 minutes.

AES

Several studies showed that AES is vulnerable to related key attacks. For example, Biryukov and Khovratovich [32] suggested a related key attack to the full AES that has $2^{99.5}$ time and data complexity. Other works [10], [24] suggested that AES is vulnerable to side-channel, chosen plain text and known plain text attacks. Alternatively, Khovratovich [33] reported biclique based attacks with complexities between $2^{124.9}$ and $2^{254.4}$ for the different versions of AES.

RSA

Since its invention, the RSA algorithm has been analyzed for security issues. However, the RSA has proven to be very secure and most of the drawbacks emerge from the result of misuse of the system, bad choice of parameters, and flaws in implementations [34]. In this sense, most of the proposed attacks exploit factoring concerns, mathematical functions, and implementation details [34]. For example, when small values of p q are selected the system can be broken using

Table 3.3: Comparative at Vulnerabilities of Classical Classical Cryptographic Algorithms.

Algorithm Name	Vulnerable to attack
DES	Brute force, Linear, Differential, Known plain text
3DES	Brute force, Linear, Differential, Known plain text, Chosen plain text
AES	Related key, Chosen plain text, Known Plain text
RSA	Brute force, Oracle attack, Side channel, Timing attacks

probability theory and side-channel attacks [29]. Likewise, to achieve a decent level of security a key length gather than 1024 buts should be used [24]. Other sets of attacks like Correlation, Coppersmith and Brute force are explained by Berlin and Dhenakaran [35]

3.2 Deep Learning

Deep learning is a complex term that accepts several definitions. However, an acceptable definition by Deng (2014) [36] is: “A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification.” Since 2006, deep structured learning, or more commonly known as deep learning, had arisen as a new field of machine learning research [37]. Over the last few years, Artificial Intelligence, Deep Learning, and Machine Learning have become well know concepts in data science. These techniques have achieved significant improvements in the field of Computer Vision, Speech and Audio Recognition, and Object Tracking [38].

The strong learning capabilities that Deep learning has have the potential to overcome conventional Machine Learning algorithms. Deep Neural Networks are constructed using the most basic building block known as the perceptron, which analyses the data in an unsupervised fashion.

3.2.1 Toward Deep Learning

Machine Learning is the technology behind modern society. It is widely applied in the search browsers, YouTube recommendation systems, e-commerce websites, and Social network recommendations. Essentially, Machine Learning systems are used to recognize multimedia content, match internet content with users, and select relevant results in Google searches. Gradually,

these applications are using deep learning techniques. Traditional Machine Learning techniques were limited in extracting features from row data representation. Over the last decades, the design of a Machine Learning system required careful engineering and expertise to design an effective feature extractor, also called a classifier, able to transform raw data into an internal representation that the system can use to effectively classify input patterns [39]. However, for many tasks, it is difficult to decide what set of features should be extracted. For example, in the design of an automobile detection system one can use the presence of a wheel as a feature. Every car has wheels. But it is difficult to express the exact representation of a wheel in terms of pixels. Since the geometric shape of the wheel is simple, it can be affected by surrounding factors like shadows, objects obstructing part of the wheel, or different degrees of brightness over the wheel.

This problem can be faced by machine learning methods that learn not only the mapping from inputs to representations but rather the representation itself. This is called *representation learning* [40]. A good example of a representation learning algorithm is an Autoencoder. It is composed of a combination of an encoder part that converts the input pattern in a different representation, unusually with less dimensional than the input; and a decoder part that transforms the new representation back into the original one. An Autoencoder is trained in such a way it preserves most of the original data as it is passing through the encoder and decoder while generating a new attractive representation. The main goal when designing a learning features algorithm is to separate the factor of variations, or sources of influence, that explain the observed data [40]. Many of the real-world Artificial Intelligence applications requires to solve the problem that every piece of data is affected by several factors of variation that can only be identified by sophisticated human understanding. Fortunately, Deep Learning emerges like a solution to this problem by introducing representations expressed in terms of other simpler ones [40]. Deep Learning algorithms are representation learning algorithms with multiple levels of representation. This multilevel architecture is obtained by composing simpler non-linear modules that transform the representation in one level into another, slightly more abstract, representation in the next level, and so on. The essential trick of Deep Learning is that this set of representations is not designed by an engineer, they are automatically generated using a general-purpose learning algorithm [39].

3.2.2 Exploring an Artificial Neural Networks

An ANN [41] is a biologically inspired computational model formed by a set of interconnected units, artificial neurons, called nodes. The multiplicative weighting factor between the input of node i and the output of node j is called the weight w_{ij} [42]. Both nodes and weights constitute the neural structure.

An ANN is an adaptive system that learns to perform a mapping between the input and output data. Adaptive means that the system parameters are changed during operation normally called the Learning/Training phase. After the training phase the Artificial Neural Network parameters are fixed and the system is deployed to solve the problem at hand [42]. Even though a single neuron is capable of performing simple classification problems, the real power comes when interconnecting several neurons in a network [43]. Often a neural network has a few hundreds of neurons while the human brain is approximately conformed by 100 billions of them. The

human brain is still outperforming an ANN in terms of creative capacity. The complexity of the human brain and its unknown mysteries is the cause of that fact. Regardless, ANN can handle huge amounts of data and surprisingly producing a high grade of accuracy. This does not make them intelligent as humans are, so computer intelligence is a better term to define this behavior. [43]

Artificial neuron

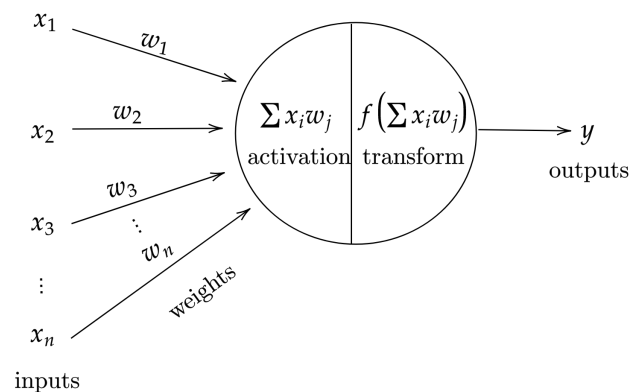


Figure 3.3: Model of an artificial neuron

In 1958, Rosenblatt introduced the mechanics of the single artificial neuron and introduced the ‘Perceptron’ to solve problems in the area of character recognition (Hecht-Nielsen, 1990) [44]. Figure 3.3 shows a schema of an artificial neuron. An artificial neuron is the building component of the ANN that mimics the behavior of a biological neuron [43]. It receives external input signals as stimuli. Those inputs multiplied by the corresponding connection weight are summed and passed to a transfer function to produce the output result. The activation function is a sum while the transfer, or transform function could be a threshold, sigmon, linear, ReLU, among others. Regardless, the most common transfer function used is the sigmoid function. Several transfer functions and their corresponding computation formulas are shown in Table 3.4 [45]:

Positive connections ($w_i > 0$) are called excitatory, they enhance the input signal and excite the neuron, while negative connections ($w_i < 0$) are called inhibitory, they reduce an inhibit the neuron activity [44].

Connection formula

Using a single neuron it is possible to solve simple linearly separable problems in which a linear hyperplane can place one class of objects on one side of the plane and the other class on the other side [44]. To face with non-linearly separable problems, extra neurons are organized in layers between the input neuron and the output layer. These intermediate layers are called hidden layers. This idea leads to multilayer architectures; Figure 3.8 shows a multilayer neural network with four input neurons, three hidden neurons, and two output neurons. Here we can identify two types of connection types, namely feedback, and feedforward. In the first

Table 3.4: Transfer Functions and their Corresponding Equations for Computation.

Function	Computation Equation
Linear	$f(x) = x$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Hard Sigmoid	$\max(0, \min(1, \frac{x+1}{2}))$
Threshold	$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$
Symmetric Treshold	$f(x) = \begin{cases} -1, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
ReLU	$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases}$

case, happens exactly what the name suggests, the output of one layer returns as the input of a previous layer. This kind of network keeps a memory of the previous state so that the new state depends on the inputs and previous states of the network. On the contrary, in feedforward networks, there are no feedback connections. The input values can only travel from the input layer to the output layer.

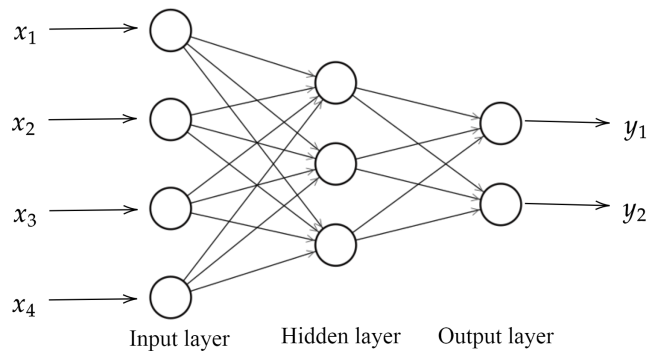


Figure 3.4: Three layer neural network

Learning rule

For the training phase, an appropriate learning rule that makes possible the mapping from inputs to output is used. The most often used learning algorithm is the Delta rule, also known as Backpropagation [43]. With this algorithm the input data is repeatedly presented to the ANN. In each iteration an error is computed based on the difference between the output values and the expected ones. This error can be defined as:

$$E = \frac{1}{2} \sum_{k=1}^p \sum_{i=1}^M (z_i(k) - y_i(k))^2, \quad (3.6)$$

where the first sum is on the p patterns of the data set and the second sum is on the M output neurons. $z_i(k)$ is the target value for output neuron i for pattern k , and $y_i(k)$ is the corresponding response output of the network. This error is feedback (backpropagated) to the input layer in such a way that the error decreases in each iteration and the ANN is closer and closer to producing the expected output values. The following formula describes the weights adjustment:

$$\Delta w_{ij}(k) = -\eta \frac{\partial E}{\partial w_{ij}(k)} = \eta [z_i(k) - y_i(k)] g'_i(h_i) s_j(k), \quad (3.7)$$

where η is the learning rate that has to be set in advance (a parameter of the algorithm), g' is the derivative of the sigmoid function and h is the synaptic potential previously defined, while the rest of the weights are modified according to similar equations by the introduction of a set of values called the “deltas” (δ), that propagate the error from the last layer into the inner ones, that are computed according to next Equation:

Table 3.5: Comparison Between Traditional Computing and ANNs.

Characteristics	Conventional computing	Artificial neural networks
Learning method	By rules	By examples
Functions	Logically	Perceptual patterns
Processing style	Sequential	Parallel

$$\delta_j^l = (S_j^l)' \sum w_{ij} \delta_i^{l+1}. \quad (3.8)$$

After the weights are properly adjusted, the system is ready to correlated unknown data.

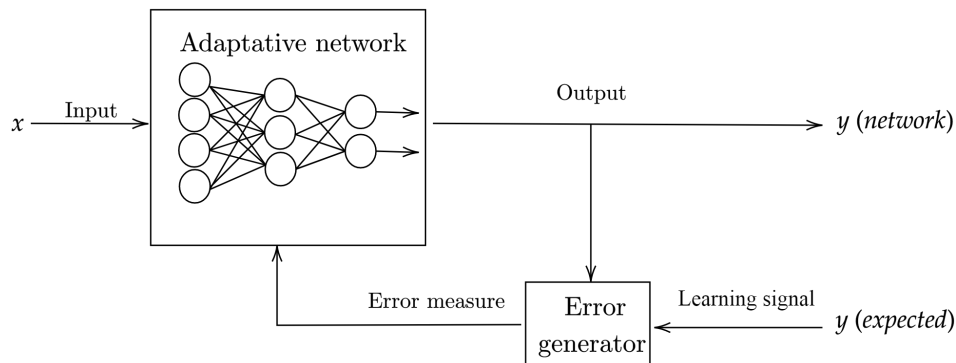


Figure 3.5: Backpropagation learning rule.

Conventional computing and artificial neural networks

A neural network is a tool designed to mimic human methods of learning and inductive knowledge by the simulation of the biological brain learning process [43]. Therefore, the way neural networks work is different from conventional computing as it is shown in Table 3.5.

3.2.3 Learning Styles

The learning process deals with finding statistical regularities or other patterns in the input data [46]. In this sense, Machine learning techniques can be classified according to the desired outcome of the algorithm. Usually, they are used for either regression, which spans around continuous data and its main goal is to make future predictions, or classification tasks, where a data set is grouped in different classes. Regardless, most of the authors propose four main learning styles. Figure 3.6 shows them with a short description of the required data.

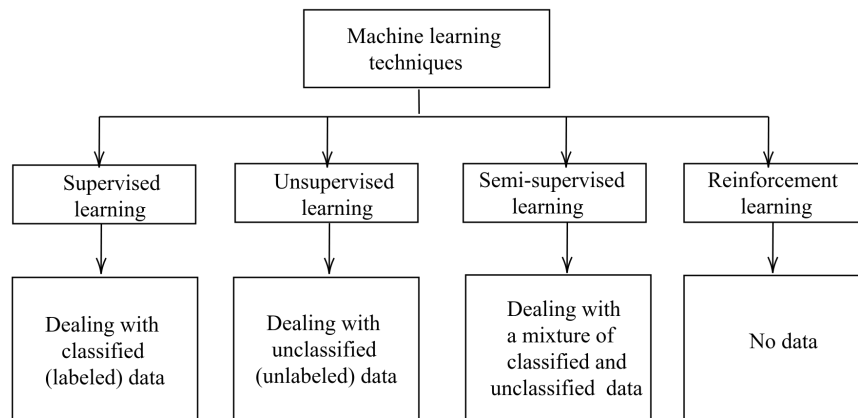


Figure 3.6: Different machine learning techniques

Supervised

In supervised learning, the goal is to infer a function from *labeled data* [47]. The training data is composed by a combination of a vector of the form $\mathbf{x} = [x_1, x_2, \dots, x_n]$, where each x_i is an *attribute* or *characteristic* of \mathbf{x} ; and its corresponding labels or *tags* in a output vector \mathbf{y} . If there is no labels for certain \mathbf{x} it is *unlabeled data*. Each pair of \mathbf{x} and \mathbf{y} vectors conforms a *training example*, *sample* or *pattern* or *instance*. A collection of training examples is known as a *training data set*. This kind of algorithm is called supervised because of the presence labels for each training example. The labels are provided by a supervisor, which usually is a human. Supervised learning is more expensive because it needs an expert that assigns the label for the raw data. Even though machines can also label data, their judgments are not as strong as a human approach.

Supervised learning is quite frequent for classification problems where the learning of a classification system by a computer is needed [46]. Such techniques are widely used in feedforward or multilayer models ML [48], like neural networks and decision trees [46]. Both of these models trust on the information given by the pre-classification of the input data. In the case of neural networks, this classification is used to minimize the error of the network by adjusting its inner weights. Alternatively, a binary tree uses the categories to define what attributes of a given pattern are more useful for classification purposes.

The very first step to train a ML is collecting the data set. In some cases an expert suggestion for how many attributes to select is really appreciated, otherwise a “brute force” method that measures everything in the hope that the right features are extracted can be used [46]. However a “brute force” does not guarantee that because of the presence of noise and missing attributes values that requires extra preprocessing [49]. The second step is data preparation and preprocessing. This face deals with the problem to handle missing data [44] and select the most significant patterns. Instance selection is a key aspect for training task which avoid large com-

putations on huge data sets. In this sense instance selection tries to minimize the data set size while maintaining as much of the mining quality of the data. Feature selection is another crucial aspect in data preprocessing that removes redundant features [50]. This procedure minimizes the dimensionality of the data and enables ML to work faster and effectively. Since the interdependence between features impact on the accuracy of ML models, a technique called *features construction/transformation* emerges as a solution for that problem. This constructs new features based on the basic feature set [46] and produces more accurate classifiers. In addition, a better understanding of the learned concept can be achieved by the selection of the most significant features.

The goal in classification problems is to minimize the error between the desired output and the actual network output. This set of examples used during the training step is called *training set* and the set of examples used to evaluate the classifier performance is known as *test set*. Usually, the entire data set is divided into 70% of examples for the training set and 30% for the test set. The agent tries to learn using the training set. However, learning the given examples is the right move. When some input data is corrupted or it is not well classified the model tends to memorize the given examples rather than develop generalization. This problem is called *over-fitting*. According to [51] the supervised ML learning algorithms can be classified in:

- Linear Classifiers
- Logistic Regression
- Naïve Bayes Classifier
- Perceptron
- Support Vector Machine
- Quadratic Classifiers
- Decision Trees
- Random Forest
- Neural Networks

Unsupervised

In unsupervised learning, the goal is to discover hidden patterns in *unlabeled data*, the idea is to learn and organize information without giving an error signal to evaluate the potential solution [48]. Here a computer must learn how to do something without telling it how to do it [46]. There are several reasons why unlabeled data appears in ML problems for example lack of investment to pay for manual labeling or the nature of the data itself [47]. In the last few years, the data collection mechanism has been significantly improved and has led to a new branch of study called *Big Data*. Most of this huge amount of information is unlabeled and today's challenge is to get something from it.

There are two approaches for unsupervised. The first one is by using a reward system to indicate success rather than an explicit categorization of the input data. This is a kind of decision problem framework because instead of producing a classification system, the agent takes a decision in such a way that maximizes the reward. This method fits real-world problems perfectly where the agent is rewarded when doing right or punished otherwise. Usually, a kind of reinforcement learning can be used for unsupervised learning [46], where the agent takes actions without learning how those actions affect the real world. In some sense, information about how an action affects the real world is unnecessary because by learning a reward function the agent always knows the expected reward for every action it could take [46]. Even though learning by trial and error is very time-consuming, this technique is useful in cases that require the computation of every possibility. The second well form of unsupervised learning is called clustering. In this approach, the objective is not to minimize an error function, but discover common patterns in the training unlabeled data [48]. Sometimes the lack of direction for the learning algorithm in unsupervised learning can be advantageous because it promotes the model to look back for patterns that have not been previously considered [52]. In this type of learning, it is assumed that the classification generated by the model will fit with an intuitive classification. For example, for demographic studies that span around the classification of people based on their economic power, the system will generate two groups: poor and wealthy. Even though the system is not fed with the group's names, it can produce them an assign one input in one or another cluster. This a data-driven approach that works fine when using enough data [46]. For example, social information filtering algorithms like the ones used for Amazon or Facebook to suggest products or new friends, are based on clustering groups of people with something in common.

According to [53], unsupervised learning algorithms are designed to extract structures from data samples. In this method, a cost function is minimized to infer the optimal parameters that characterize the hidden structures in the data [46]. On the one hand, a robust inference is the one that is able to extract the typical characteristics of the data source. In other words, similar structures should be extracted from the second set of samples from the same data source. On the other hand, lack of robustness is known as over-fitting. In some sense, unsupervised learning find structures in the data beyond what would be considered as pure unstructured noise [54].

Examples of unsupervised learning algorithms are [46]:

- K-means
- Gaussian Mixture Model
- Self-organizing Maps
- Principal Component Analysis
- Hidden Markov Model

Semi-Supervised

Semi-supervised learning is halfway between supervised and unsupervised learning [55]. This type of learning combines labeled and unlabeled data. Usually, the amount of unlabeled data

is greater than the amount of labeled data [47]. These problems appear in environments where collecting data is cheap but labeling it is very expensive and time-consuming. The key is to identify how the distribution of the unlabeled data affects the supervised learning [56]. The goal of semi-supervised learning is to produce a model that will predict classes of samples better than the model generated by using the labeled data alone [47]. In transductive learning, the model is aside from the fact that the interest is only in the predictions on the unlabeled training data, without any intention to generalize [49]. Another setting for semi-supervised learning is by using constraints [55]. For example, there can be a constraint that binds samples to the same target. This approach is seen as unsupervised learning guided by constraints.

Semi-supervised learning is valuable by its practical value in learning better, faster, and cheaper [57]. Nowadays, it is relatively easy to collect large quantities of unlabeled data. For instance, documents can be downloaded from the Internet, images can be taken using smartphones, and medical images can be retrieved from hospitals. However, assigning their corresponding labels usually requires an expert supervisor and several hours of work. This fact is the main reason why there is an excess of unlabeled data and limited labeled data. Therefore, semi-supervised learning emerges like an opportunity to take advantage of the excess of unlabeled data. In recent times, semi-supervised learning has been in cognitive psychology as a computation model for human beings [57]. In human reasoning and categorization, nature provides unsupervised data along with labeled data. For instance, a child spontaneously seeing around a window and his father saying “hey, that is a bird”. Several studies have suggested that human beings combine labeled and unlabeled data to facilitate learning. Examples of semi-supervised learning algorithms are [57]:

- Generative Models
- Semi-Supervised Support Vector Machines
- Graph-Based Models
- Co-training and Multiview Models

Reinforcement

In reinforcement learning the machine aims to produce a set of actions to interact with its environment. The actions produce a response in the environment and reward or punish the machine with a scalar value [53]. Thus, the goal of reinforcement learning is to maximize the rewarding signal. A decision-making function is used to make the agent take an action and the information about the received reward from the environment is stored to develop a policy of how to act given a particular observation of the real world [46].

A key aspect of reinforcement learning is the equilibrium between exploration and exploitation [58]. To maximize the reward, the system usually tends to utilize actions it tried in the past that produced a successful result but to discover such actions the system has to try actions that it has not tried before. In this sense, the system has to *exploit* what it already knows, but also *explore* new options to produce an even better reward. However, the thing is that neither exploration nor exploitation can be effectively isolated [58].

An interesting fact about reinforcement learning is its strong interaction with other scientific and engineer disciplines. Since the early stages of artificial intelligence and machine learning, reinforcement learning has been used like a bridge to statistics, optimization, and other mathematical subjects. For example, in research and control theory, reinforcement learning methods can be applied to learn from parameterized approximators [58] [54]. In addition, reinforcement learning has produced interesting results in the area of psychology and neuroscience. Most of the essence of reinforcement learning algorithms were originally inspired by biological systems. Furthermore, reinforcement learning is the closest to biological intelligence [58]. Some of the well known reinforcement learning algorithms [59], [60] are:

- Q -learning
- Dynamic Programming
- Value Iteration
- Advantage Learning
- Temporal Difference Learning

3.2.4 The Architecture of Neural Networks

Real world problems need strong power of processing that a single neuron, even with multiple inputs cant solve. It is necessarily to have several neurons working in parallel organized in *layers*. In this sense, a neural network has three main components or layers:

- *Input layer*: This layer receives the input information, i.e data, features or samples, from the environment. Usually, this input information is normalized withing the range of the activation function to have better network performance [61].
- *Hidden layer*: Inside this layer occurs the necessary processing for extracting features from the input data.
- *Output layer*: This layer shows the final output of the network resulting from the processing of the hidden layer.

Artificial neural network architectures can be classified according to the neurons arrangement and how they interoperate. This categorization includes [61], [62], [63]: (i) single-layer feed-forward , (ii) multilayer feedforward networks, (iii) recurrent networks and (iv) mesh networks

Single-layer Feedforward Architecture

This kind of neural network has only one layer of neurons, which actually is the output layer [61]. Figure 3.7 shows a single layer neural network with n inputs and m outputs. Every input x_i is connected with all the neurons through a weight matrix W :

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

The information always flows from the input layer to the output layer, it is *unidirectional*. Single-layer feedforward networks are widely used for pattern classification and linear filtering problems [61]. A well know examples of this architecture are the perceptron and Adaline [64] which uses the *Delta Rule* and *Hebb's Rule* for the training process.

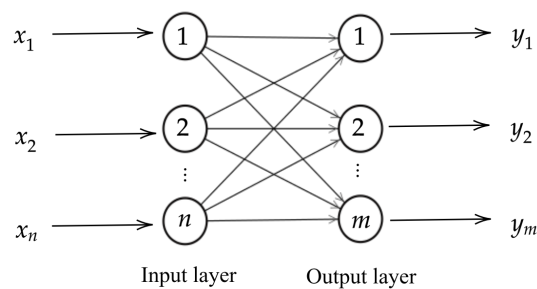


Figure 3.7: Single-layer feedforward network

Multi-layer Feedforward Architecture

In a different way, a multilayer feedforward network consist of one or more hidden layer [62]. Each layer of neurons has its own weight matrix \mathbf{W} . This type of architecture is applied to solve complex problems that require a strong processing power like pattern classification, function approximation, optimization, robotics, system identification, among others [61]. In this sense, this kind of networks are more powerful and versatile that single-layer networks. For example, having one layer with a sigmoid activation function and another with a linear activation function can approximate a function pretty good [62].

In Figure 3.8 is presented a multilayer feedforward network that receives an input vector of dimension n ; and is composed by two hidden layers with n_1 and n_2 units, respectively; and one output layer with m units. A key aspect of this architecture is the fact that the number of neurons in the hidden layer tends to be different from the number of neurons in the input layer. This is because the arrangement of the hidden layers depends on the nature of the problem being solved by the network as well as the amount and quality of the available input data [61]. Although the number of neurons for the hidden layers as well as how many of them should be used for a specific problem is sometimes determined by trial and error, dealing with the selection of the

optimal number of neurons for the hidden layer is a problem that is faced for an entire area of research. [62]

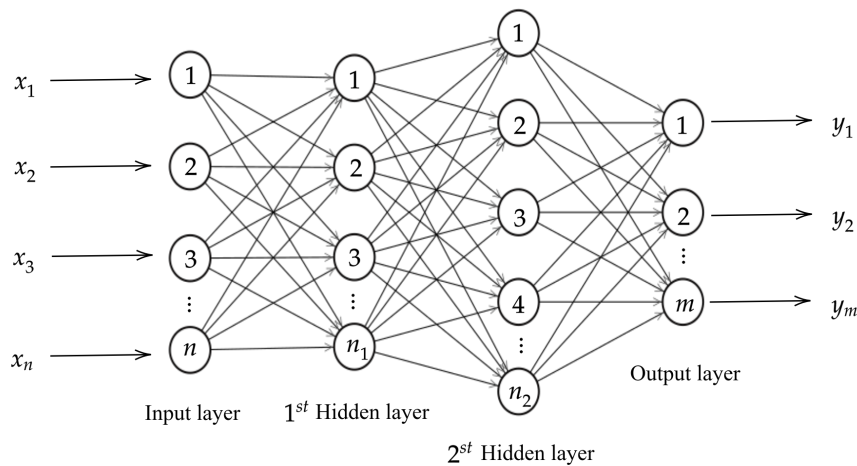


Figure 3.8: Multilayer Feedforward Network.

Nevertheless, the number of neurons for the input and output layer is specified by the characteristics of the training data set. For example, in the development of a classification system, let's say a cancer disease detection system. The input vector would consist of n features that will be mapped to two classes: presence or absence. Therefore, the number of neurons for the input and output layers are set to n and 2, respectively.

Recurrent or Feedback Architecture

In a recurrent neural network, some values from the output layer are used as input, i.e. *feedback* connection, for a previous layer of neurons [62]. In this way, current output values depend on the previous ones. This architecture is quite different from the preceding ones which were unidirectional with no backward connections. The feedback particularity of these networks enables them to be ideal for dynamic information processing like time series prediction, process control, system optimization and organization, and so on [61]. In addition, this capability produces a *temporal behaviour* and *sequence learning* that gives to the network a sort of *memory* [65]. Moreover, the feedback loop implies the use of special branches constituted by *unit-delay elements*, which produces non-linear dynamical behavior when using non-linear neurons [66]. Typical examples of this type of architectures are Hopfield neural networks and the Perceptron with feedback whose learning algorithms are based on energy function minimization and the generalized delta rule, respectively [64].

Figure 3.9 shows an example of Perceptron with feedback where one of the output values is backpropagated to the middle layer.

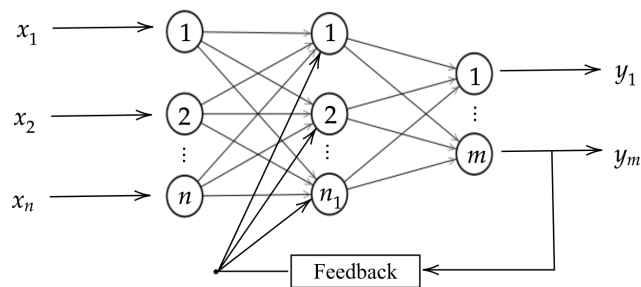


Figure 3.9: Recurrent Network.

Mesh Architectures

Mesh architectures are characterized for considering the arrangement of the neurons for pattern classification problems based on how the synaptic weights and threshold will be adjusted. Furthermore, mesh models have achieved important results for data clustering, pattern recognition, system optimization, among others. [61]. A well-known example of a mesh architecture is the Kohonen network, which is trained in a competitive way. Figure 3.10 presents an example of a Kohonen network where its hidden units are organized in a two-dimensional space. In this architecture, the input signal is spread to all the neurons in the network. In addition, Kohonen neural networks are a type of *self-organizing neural networks*, which can be arranged as a grid and uses a *neighborhood function* to perform clustering operations [67]. During the training process, the weights and neighborhood function are updated in such a way that closely spacial located neurons react to closely related inputs

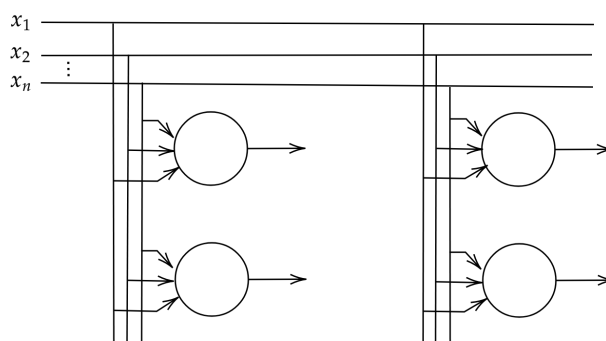


Figure 3.10: Mesh Network.

Autoencoder Architecture

The architecture of an artificial neural network with Autoencoder architecture is composed of a first layer in which the input data is presented and an output layer in which the presented data is recovered. Both layers have to be implemented with an equal number of neurons units (neurons), and one or more hidden layers. [68]. In addition, in order to use the Autoencoder for encryption/encryption purposes, the architecture must be symmetric. An Autoencoder is trained in an unsupervised way using backpropagation using the inputs values as its targets. Therefore, what an Autoencoder does is try to learn a function $h_{W,b}(x) \approx x$. In other words, given an input vector, it tries to learn how to reconstruct it at the output layer such that x very similar to \hat{x} . The architecture for an autoencoder is composed of three hidden layers with m, h and m neurons respectively, and receives a n -dimensional vector is shown in the Figure. 3.11.

Although, the approximation if the identity function seems not much interesting, an autoencoder can discover fascinating structures on the input data when imposing constraints on the number of hidden neurons [69]. For instance, assume an image compression system. Assume that the inputs are the pixel values of an 10x10 image, thus $n = 100$ and $x \in \mathbb{R}^{100}$ and there are 50 neuron in only one hidden layer. In this case, the system is forced to first compress the input data from \mathbb{R}^{100} to \mathbb{R}^{50} and then reconstruct it to \mathbb{R}^{100} again. The compression procedures would be very difficult when working whit independent and identically distributed (IDD) random variables. However, if some patterns are in the data, this algorithm will discover it [69]. Sometimes, compressing the input is not the only way to discover features in the data set. Even if other constraints are in the hidden layers like a large number of hidden units, the autoencoder can still discover interesting structures, this is the concept of *sparsity*.

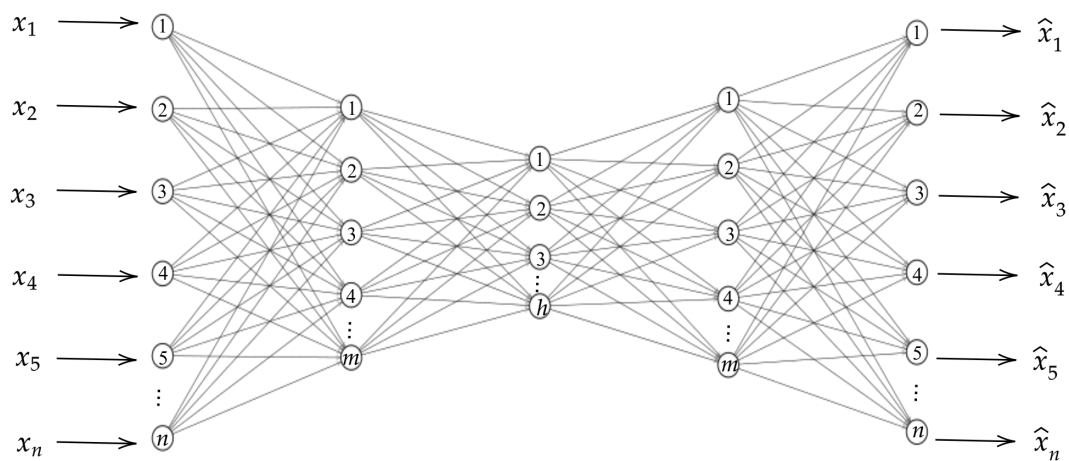


Figure 3.11: Architecture of an Autoencoder

Autoencoder was introduced for the first time in the 1980s by Hinton and the PDP group [70] to overcome the problem of *backpropagation without a teacher* by employing the input data as the teacher [71]. Moreover, in the last years, autoencoders have gained an important status in deep learning approaches [72], [72], [73] were the layers of an autoencoder are trained separately in an unsupervised way and then stacked to fine-tune the entire network to form a

Restricted Boltzmann Machine. This network arrangement has shown relevant improvements of several state-of-the-art problems in classification and regression [[71](#)]

Chapter 4

State of the Art Review in Cryptography with Deep Learning

So far, it has been discussed that deep learning is a subarea of machine learning and an application of ANNs. There, state of the art in cryptography with deep learning spans around the use of ANNs for cryptography tasks. This section presents some previous research related deep learning and cryptography focused on ANNs such as neural cryptography, pseudo random number generators, visual cryptography, deep learning attacks, and then focuses on previous research about DNA cryptography and deep learning.

4.0.1 Neural cryptography

Several researchers such as Jogdand and Bisalapur [74], and Kinnzel and Kanter [75] have studied key management, generation, and exchange based on mutually learning neural networks. They proposed a neural key exchange mechanism based on the synchronization of two parity machines, which are a kind of feedforward neural network composed by one output neuron, K hidden neurons and $K*N$ input neurons.

For example, [74] is proposed that neural networks can be used as a secure key exchange mechanism. In this approach, two parity machines, one for the sender and receiver side respectively, receive the same input vector, generate an output bit, and are trained based on the output bit. It seems that two neural networks mutually trained, achieve a synchronization state where their time-dependent synaptic weights are exactly the same. The hidden values are equal to the sign function of the dot product of input and weights while the output value is the multiplication of hidden values. The training process is carried out comparing the outputs of the corresponding tree parity machines of two partners A and B. Finally, the generated secret key over a public channel is used for both encryption and decryption of data using traditional techniques like AES [76].

Likewise, [75] stated that it has not been proved the absence of an algorithm for success attack, but this approach is very hard to crack by brute force. Even though an attacker knows the input/output relation and knows the algorithm, he is not able to recover the secret common key that A and B use for encryption. Neural networks are the unique algorithm for key generation over a public channel that is not based on number theory. Its main advantages over traditional

approaches are: it simple, low computations for training, and a new key is generated for each message exchange.

In addition, simple neural networks have also been used for the generation of a secure cryptographic secret-key using a public channel. For instance, Klein et al. [77] described the learning process where two perceptrons receive a common input and change their weights according to their mutual output. Furthermore, they suggested that a new technique that combines neural networks with chaos synchronization seems to be the most secure against attackers. Finally, this works explains a different kind of attacker techniques and suggest that using a big enough weight space the systems becomes more secure.

Although this methodology is secure in the sense that when both parties are synchronized, an attacker using a similar neural network is unlikely to converge to the same generated key, three ways to brake this exchange protocol were proposed in [78]. The first one is based on genetic algorithms. In this approach, an initial population of a neural network is simulated with the same structure as the two parity machines. After each round of training half of the population gets the same output as the target parity machines. Then, networks whose outputs coincide with them evolve and multiply. Finally, the attacker checks if one of the networks inside the population has the same weights that one of the parties. The experimental test achieved 50% of effective attacks. The second type of attack exploits the geometric interpretations of the actions of a perceptron to discover the hidden weights of the parties. Here the attacker tries random initializes a network C and tries to correct the punctual hidden outputs that are different from the synchronized parities A and B by analyzing their outputs on a common input vector x . The last attack is a probabilistic approach to detect the perception being updated in each round as well as estimate the probability distributions of the synaptic weights. Regardless, the proposed method in this work is not affected by this kind of problem since it does not deal with key exchange protocols but actually encrypts and decrypts data similar to the approach proposed by Volna et al. [79]. In this study, two neural networks are trained for encryption and decryption respectively. In this approach, the key is considered as tuple composed by the network parameters.

4.0.2 Pseudo random generators

It is a fact that randomness increases the security of cryptosystems [80]. The finding that neural networks are suitable for pseudorandom number generation in the implementation of stream cipher cryptosystems is supported by Chan and Cheng [81] who presented a pseudorandom number generator based on a *clipped Hopfield* neural network and a *Linear feedback shift register* (LFSRS). Chan and Cheng also stated that since the inherent linearity of the LFSRs makes it vulnerable when observing a number of the consecutive sequences, neural networks are proposed as a non-linearity function that takes shift register vectors as inputs to crush the algebraic and linear structure of the original sequence. Karras and Zorkadies [82] who conducted a study focused on strengthening pseudorandom bit sequence generation for secure electronic commerce transactions using neural network techniques, came to similar conclusions. In their study, Multilayer Perceptrons and Hopfield architectures are proposed to evaluate and improve traditional generators. ANN can approximate pretty well nonlinear mappings whit fewer parameters than competitive approaches and can also be parallelizable. They exploit the overfitting

problem as a mechanism to enhance pseudorandom sequence generation and conclude that a neural network-based approach behaves significantly better in terms of unpredictability.

In summary, a neural network can be used either as a separate random generator or to increase the strength of up-to-date generators by taking its linearly outputs as inputs for the network [80].

4.0.3 Neural attacks

Deep learning offers attractive capabilities to brake current existing cryptography systems. A recent line of works has investigating cryptanalytic attacks based on deep learning. For example, Maghrebi et.al [83] have presented a deep learning approach for profiling side-channel attacks which is able to brake both protected and unprotected AES implementations. In particular, they used key recovery capabilities based on feature extraction procedures and exploiting time dependency between samples. Several tests were conducted on three different datasets to compare the proposed method with machine learning and template attacks. In a similar way, several attacks based on algebraic aspects have been proposed since the adaption of DES and 3-DES. However, a neural network methodology is proposed by Alani [30]. This study performed a known plain text attack that can break DES in 51 minutes and requires 2^{11} plain/cipher text pairs. For 3-DES the time is 72 minutes with a requirement of 2^{12} plain/cipher text pairs. The methodology used for the researcher in mention is simple and effective. A neural network is trained to map, without knowing the encryption key, some known plain/cipher text samples collected by the attacker.

In extension to the work proposed by Alani, a similar study but with the intention to retrieve the encryption key, instead of training a neural network to map plain/cipher text pairs, was implemented by Jayachandiran [84]. The proposed system was tested on a one-round and two-round version of Simon cipher. The election of the cipher was made with the intention to use a lightweight cipher that would be easier for the network to analyze. Another study targeting Speck32/64 is explained by [85]. This work takes advantage of the differential properties of round-reduced Speck. A neural network is trained to recognize Speck outputs. To show the better overall performance of the proposed neural distinguisher, simple key recovery attacks were constructed varying the number of rounds for Speck 32/64. Just a couple of minutes are required to train the network.

4.0.4 Visual cryptography

Cryptography with deep learning techniques not only has dealt with encryption and decryption of messages but also there is evidence of cryptography systems focused on visual content. In this way, steganalysis is a technique that allows hiding information in images. The main goal of steganography is to guarantee an attacker can not get the secret message nor distinguish the images carrying secret information [80]. Shaohui, Hongxun, and Wen [86] proposed a method to detect whether an image contains hidden or not. Firstly, they changed the domain of the object image by applying transformations like a discrete Fourier transform (DFT), discrete cosine transform (DCT) and discrete wavelet transform (DWT). Then statistical features are extracted. Shaohui et al. found that the image's statistical features are significant hints to

decide whether hiding information or not from the detection process. A similar study but using a different set of image features were developed by Shi et al. [87]. In this art, neural networks were used as an image classifier and it was found a direct relationship between the synaptic weights and an image hiding data.

A more recent study based on a customized Convolutional Neural Network (CNN) was prepared by Qian, Dong, Wang & Tan [88]. CNN is one of the most famous models in deep learning. Its name comes from the fact that it is based on the mathematical convolution matrix operator. CNN is a hierarchical neural network that is composed of several layers: convolutional layer, non-linearity layer, pooling layer, and fully connected layer [89]. This type of neural network is capable to learn complex hierarchical images' features representations by alternating application of trainable filters and pooling operations [88]. The main goal of Qian et al. was to automate the feature representation learning procedure. To accomplish that, they designed and trained a customize CNN, with several convolutional layers, to learn complex dependencies helpful for steganography. They found that the suggested model outperforms current spacial domain steganographic algorithms like HUGO, WOW, and S-UNIWARD. Another CNN based steganalysis technique for digital images was designed by Ye, Ni & Yi [90]. They found that the designed CNN is able to learn a hierarchical representation of the data and optimize key steps in steganographic detection. In the same way that Qian et al. have done, Ye et al. compare the proposed model with three up-to-date steganographic algorithms, namely: WOW, S-UNIWARD, and HILL.

Other than aspects related to CNN for steganography, a stacked autoencoder can also be perceived as suitable to encrypt images according to the findings of some researches, as presented below. Asian researches Hu, Wang, Xu, Pu & Peng [91] who designed a stacked autoencoder for bath image encryption. In spite of the improvements in communication systems developed so far, images transmitted over public channels can always be intercepted and manipulated by eavesdroppers. It is mandatory to encrypt images carrying hidden information before sending int over public channels. However, traditional cryptography schemes like AES or DES are exclusively designed for text encryption and are not appropriate for image encryption [92]. Nevertheless, several image encryption based on DNA cryptography, transform domain, mathematical concepts, and compression methodology have been proposed, but they still have security weaknesses. Furthermore, if encrypted images are cracked, the rest becomes vulnerable too. For that reason, Hu et al. proposed a batch image encryption algorithm that uses a stacked autoencoder to produce two chaotic matrices to shuffle image pixels and confuse the inter-pixels relationships. They found that the proposed scheme is resistant to a brute-force attack, statistical attack, and differential attack.

Digital watermarking is another well-known technique to protect images against attackers that have introduced the use of deep learning as a novel approach. In image watermarking, information is embedded in an image. Then, the watermark can be detected and extracted to make assertion over the transmitted image Four are the main requirements for watermarking systems: Domain watermark insertion, Watermark detection and extraction, Watermark resistance to attacks and Watermark visibility [80]. A review of the different techniques for embedding and extracting a watermark in a digital image was shown by Mohananthini1 and Yumana [93], who found that although different types of neural networks have been using for watermarking process in spatial and frequency domain, Backpropagation neural networks got better performance

results.

4.0.5 DNA cryptography and Deep Learning

DNA cryptography is a new scheme to improve information security concerns. This novel approach is inspired by the fact that DNA molecules store, process, and transmit information. Therefore, DNA cryptography is the combination of the classical cryptography techniques with the chemical inherent characteristics of DNA molecules [94]. Basically, in DNA cryptography the information is encoded in nucleotide basis (A,C,G,T), i.e it uses DNA computational techniques [95]. This can handle huge amounts of information in less time compared with traditional schemes since one cubic decimeter of DNA solution can store trillions of bits [96]. One effort to merge this DNA cryptography with deep learning was proposed by Kalsi, Kaur & Chnag [97]. In that study, a key generator was implemented using genetic algorithms. The encryption and decryption process was carried out using DNA computing. Finally, the entire system was reinforced using deep learning techniques to encrypt and decrypt data in DNA sequence format.

Likewise, a neural network key generator for using along with DNA cryptography was designed by Basu, Karupiah, Nasipuri, Halder & Radhakrishnan [98]. Specifically, a Bidirectional Associative Neural Network was trained on randomized information to produce a key, which is able to reduce memory space by recurrent reconstruction of the set of keys. Later, the data was converted from binary to DNA bases and the encryption and decryption processes mimic the natural processes of Transcription and Translation as well as the reverse processes. They found that changes 1 and 2-bit changes in the plain text produce 93.59% and 96.15% of changes in bits of the cipher text, respectively. They claimed that their system is strong enough to brute force, cipher text only known plain text and cryptanalysis attacks.

An effort to combine neural networks, chaotic systems, and DNA computing for image encryption was studied by Maddodi, Awad A., Awad D., Awad M. & Lee [99]. They proposed a method where a four-layers neural network and chaotic methods are used for pseudorandom sequence generation which is used as inputs to the proposed DNA computing based cryptography algorithm. Pixel substitution and permutation were controlled using the pseudorandom sequence. By the execution of several tests and comparisons with similar works, Maddodi et al. proved that the proposed system has high cryptography quality.

Chapter 5

Methodology

This section describes the methodology used to implement a symmetric key cryptography system using deep learning. This research focuses on the development of a low computational time consuming and secure cryptography algorithm. As previously discussed in this work, a sparse autoencoder neural network architecture was selected for encryption and decryption of data. Additionally, a method to make the functional output of the autoencoder(i.e. the synaptic weights) dependable on a secret piece of information (i.e. the key) was designed.

5.1 Research Approach

In this work, we developed an experimental study to design a robust and efficient symmetric key cryptography system based on deep learning techniques. This research was conducted in two parts: (1) The calibration of an autoencoder neural network to make it able to encrypt and decrypt the complete set of ASCII character, (2) A performance study and theoretical security analysis to prove the system is secure against attacks.

5.2 Encryption and decryption approach

This section describes in detail the encryption and encryption process when interchanging messages.

A flowchart of the encryption process is presented in Figure 5.1. First, every character of the plain text is transformed into its corresponding 8-bits ASCII representation.

For example the text “Hello” is encoded into binary ASCII sequences as:

$$P = 0100100001100101011011000110110001101111$$

After converting sequence into binary form the sequence is divided in to blocks of 8-bit each:

$$P' = 01001000 \ 01100101 \ 01101100 \ 01101100 \ 01101111$$

Then, every 8-bit set passes through the hidden weights W_E , and the outputs of the 9-neurons hidden layer compose the encrypted version of each 8-bit input set. Since the range of the activation function of the hidden layer is $[0, 1]$, its outputs are floating-point numbers inside that range. Finally, the cipher text is composed by concatenating every 9-dimensional vector to

form an array of floating points numbers which has a length of 9 multiplied by the number of characters the plain text has:

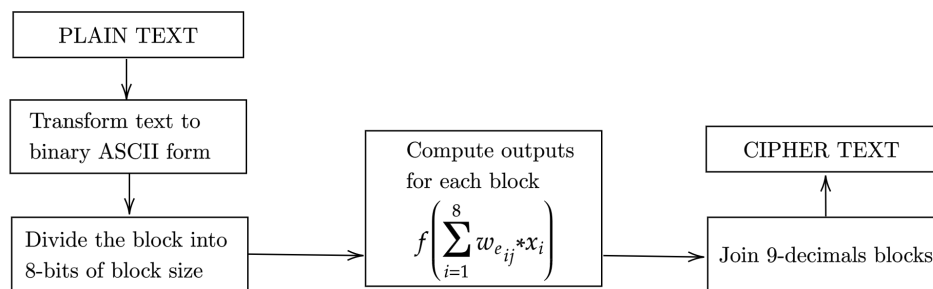
$$C = 0.999846 \ 0.882371 \ 0.220441 \ 0.000862 \ 0.870736 \ 0.369639 \ 0.002191 \ 0.491044 \ 0.542451 \\ 0.992447 \ 0.070593 \ 0.000766 \ 0.992716 \ 0.156359 \ 0.549780 \ 0.549430 \ 0.500320 \ 0.995227 \ 0.999920 \\ 0.907750 \ 0.000229 \ 0.049527 \ 0.617286 \ 0.029654 \ 0.452724 \ 0.598132 \ 0.507328 \ 0.999920 \ 0.907750 \\ 0.000229 \ 0.049527 \ 0.617286 \ 0.029654 \ 0.452724 \ 0.598132 \ 0.507328$$


Figure 5.1: Encryption Process.

The decryption process is analogous to the encryption process and it is depicted in Figure 5.2. First, the receiver splits the received array into blocks of 9 floating-point numbers and process them with the weight matrix of the output layer \mathbf{W}_D . After every block is processed, every 8-dimensional vector of floating-point numbers produced in the output layer is discretized by applying a threshold function, see equation 5.1, and transformed again to a character using the ASCII table. Finally, the plain text is obtained by the concatenation of the obtained characters.

$$\phi(x) = \begin{cases} 0, & \text{if } x < 0.5 \\ 1, & \text{if } x \geq 0.5 \end{cases} \quad (5.1)$$

It is important to point out that trough this approach every party involved in the communication can encrypt and decrypt data. Likewise, a different password can be used every time a conversation starts or set a new one after a given amount of time.

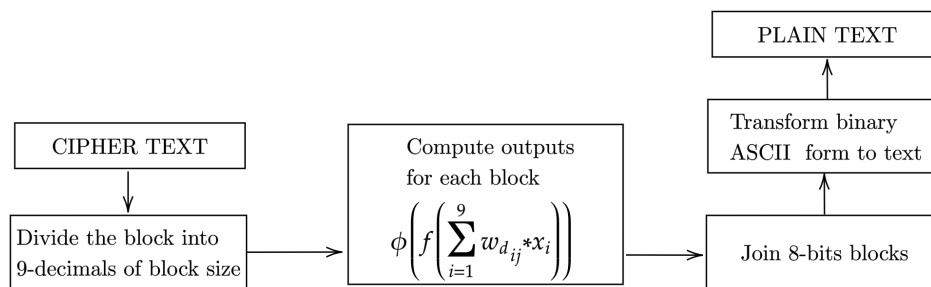


Figure 5.2: Decryption process.

5.3 Symmetric Key Cryptography Model

In this thesis, a new symmetric-key encryption system using an autoencoder architecture is proposed. This approach combines: (1) A sparse autoencoder to encrypt and decrypt the data, (2) A key generator to guide its learning process. The proposed system is symmetric in the sense that the key generator transforms a secret alphanumeric password, only shared by the two parties involved in a communication process, to a cryptographic key. The key is a sequence of numbers that will be fixed as the seeds of every random process required to train the ANN. In other words, the user login password is used to determine the learning process and therefore the final state of the network

The system works as follows: First of all, both the sender and receiver (i.e. the users) are provided with the same autoencoder and secret login password. Next, the system is initialized. This process encompasses two steps: (1) Key generation using the login password, and (2) Initialization and training of the autoencoder using backpropagation. Once the network is capable to codify and retrieve the entire set of ASCII characters in binary format, the system is ready to transmit data over public channels. A depiction of the proposed model is presented in Figure 5.3.

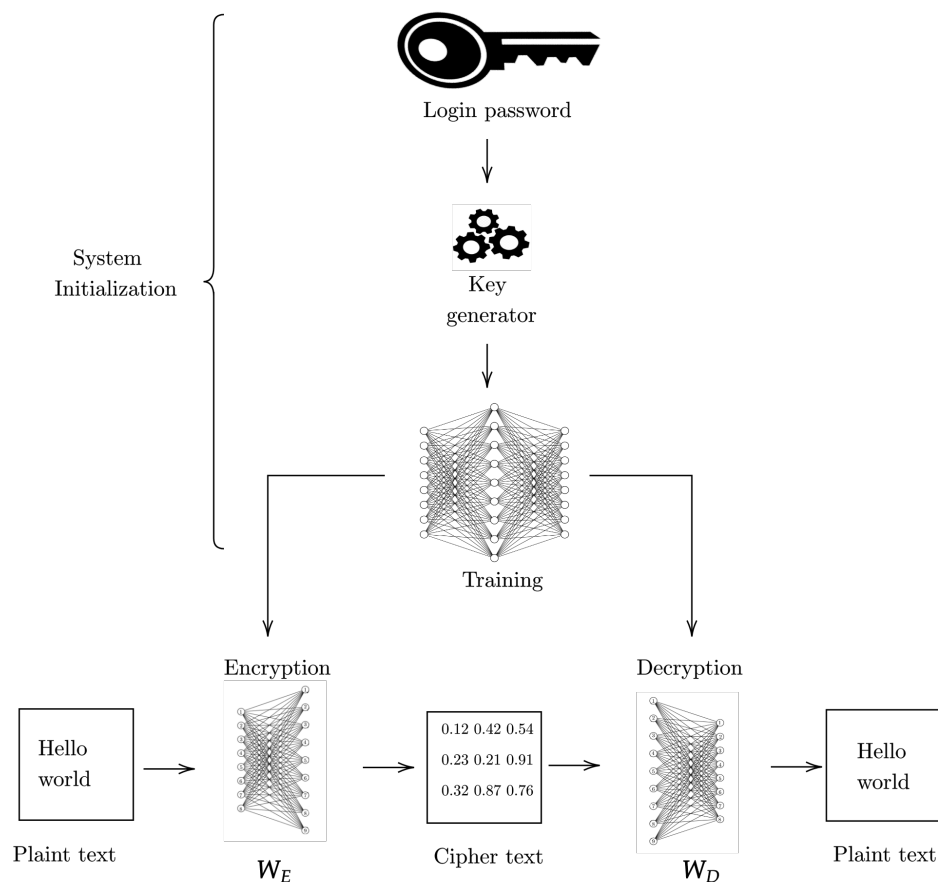


Figure 5.3: Overall flowchart of the proposed symmetric key cryptography system.

To fulfill the cryptography laws, both components of the proposed systems the architecture of the autoencoder, as well as the key generator algorithm, are publicly available. Their designs are discussed in the following subsections. To sum up, the proposed system requires a private password shared by the sender and receiver while its structure is public.

5.3.1 Key generator

It is a fact that the final synaptic weights produced after the training process depend on the initial random synaptic weights generated by pseudorandom number generators. Most pseudorandom number generators are built on algorithms involving recursion over a base value called the “seed”. In this sense, the same seeds produce the same sequences of random values. Therefore, if the same seed is used to initialize the synaptic weights of two separate autoencoders, after the training process their tuned synaptic weights will be exactly the same. This effect is exploited in this research to make the replicability of the autoencoder relies on a sequence of seeds, which will be considered as the key. Therefore, a key generation algorithm that transforms an

alphanumeric password into a sequence of integer numbers, which are used as the seeds in the random processes of the training stage, was designed to enforce the security of the proposed system.

In this implementation, the generated key has the same length of the login password and every call to a random process uses a different seed. Due to the training process requires more seeds, the value of seed is increased each time after used to avoid repeated seeds in the next search inside the seeds array. First, several rounds over the seeds array are performed until set all the initial synaptic weights of the autoencoder, in this process, 144 seeds are required. Admittedly, the order in which the inputs are presented to the network affects how the network generates internal representation. This fact is exploited to galvanize the system by performing next rounds over the seeds array to randomly shuffle the training dataset each epoch of the training step, an average of 14 seeds are required in this stage. By randomly altering the order in which the information is presented to the network, a chaotic environment that adds entropy, and therefore strength to the proposed system is encouraged. Figure 5.4. shows a depiction of the key generation algorithm proposed in this work.

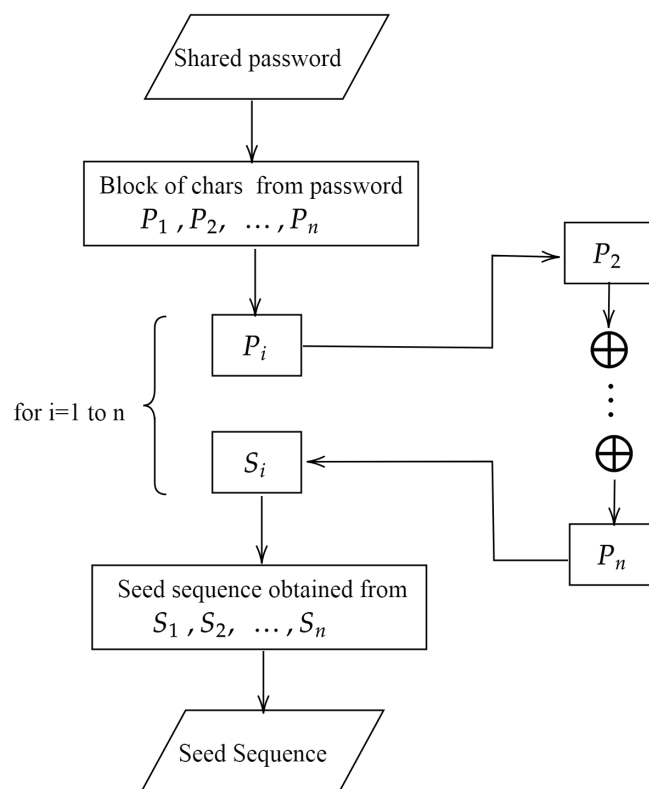


Figure 5.4: Seed Sequence Generation Flowchart

5.3.2 Autoencoder architecture

A sparse autoencoder with 8 neurons in the input layer, 9 neurons in the hidden layer, and 8 neurons in the output layer was selected. The reason why a single layer was chosen is that a single hidden layer keeps the system simple and powerful enough when more neurons than the input layer are added. Several experiments were carried out to fine-tune the network hyperparameters that makes the network able to encrypt and decrypt the information in a reasonable amount of time.

The activation function for this implementation is a sigmoid function with a gain value: β according to the following formula:

$$f(x) = \frac{1}{1 + e^{-\beta x}} \quad (5.2)$$

Having a variable gain value allows having layers with different versions of the sigmoid function. Such a mechanism allows smoothing the effect of the gradient descent problem. The ANN has many user-supplied parameters whose values are fixed before the learning process begins known as *hyperparameters*. They are heuristically chosen and experimentally verified. In particular, the hidden layer uses a gain value of 10.5 while the output layer has a gain value equal to 5.5, which was selected after several experiments. The learning rate selected was 0.25 and the interval for the initial synaptic weight was fixed to [-0.5,0.5]. Table 5.1 summarizes the hyperparameters of the sparse autoencoder and a graphical representation is shown in Figure 5.5

Table 5.1: Hyperparameters List of the Proposed Neural Network

	Parameter	Value
Autoencoder	Training Algorithm	Backpropagation
	Activation Function	Sigmoid
	Number of Neurons	8-9-8
	Gain Values	10.5-5.5
	Initial Weight Range	[-0.5-0.5]
	Learning Rate	0.25

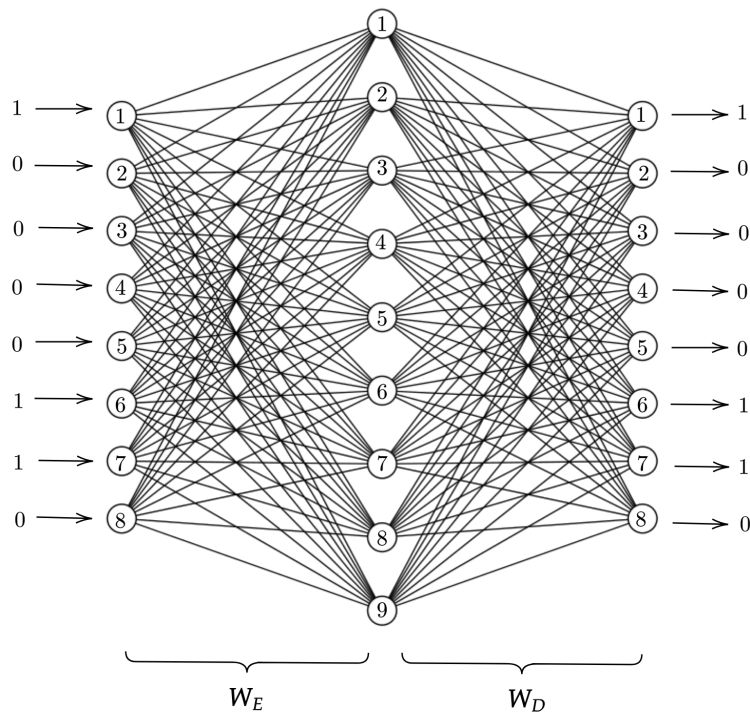


Figure 5.5: Autoencoder Architecture of the Proposed Model

Chapter 6

Experimental Setup

In this Chapter, the experimental setup to evaluate the performance and security level of the proposed model is described. Performance evaluation parameters were selected and compared with classical cryptographic algorithms, namely AES, DES, 3DES, and RSA. Additionally, experiments to estimate how feasible breaking the system by a brute force approach were conducted. Since every symmetric encryption algorithm is open, in the proposed system an eavesdropper has prior knowledge of the architecture of the network and its hyperparameters as well as the key generation algorithm. In this sense, an attacker trying to break the proposed system desires to get the weights of the network. The way one can do that is by trial and error.

6.1 Software and Hardware

The proposed system was implemented in C++ with Borland compiler using a machine under Windows 8.1 64 bits operating system, 4Gb DDR3 RAM and Intel i5-5200U 2.20GHz CPU

6.2 Performance Evaluation Parameters

To apply a specific algorithm to an application, a clear understanding of its performance parameters is required. In this work, analysis of the following metrics is done to compare the performance of the proposed system between different cryptographic algorithms.

6.2.1 Encryption time

The time that is taken to get the cipher text from the plain text is the encryption time. Encryption time depends on the key size, plain text size, and mode. In this work, the encryption time is measured in milliseconds. The complete system performance is directly affected by the encryption time. Ideally, the encryption time should be short to guarantee speed and responsiveness.

6.2.2 Decryption time

The time taken to retrieve the plain text from the cipher text is the decryption time. To guaranty speed and responsiveness, decryption time should be less or similar to the encryption time. In this work, decryption time is measured in milliseconds.

6.3 Security Analysis Parameters

To develop the security analysis of the proposed system, a couple of parameters are established.

6.3.1 Accuracy

The number of ASCII characters that are retrieved over the total data set is the accuracy. It is desired that the performance of the system decreases when altering all the weights of the network. In this work, the accuracy of the network is measured in percentages.

6.3.2 Sensibility of the network

The sensibility of the network is a representation of the accuracy of the network whilst the noise level inside the synaptic weights is increased.

6.3.3 Cut-off point

The synaptic weights of the ANN belong to a continuous domain and hackers must try infinite numbers in this domain to reach the right synaptic weights. This is not practical and therefore a way to discretize the weights space is proposed in this work by fixing a decimal unit that splits this range. This quantity is defined as the *cut-off point*.

For this work, the cut-off point is the decimal unit that decreases the accuracy to 50%. It can be, tenths, hundredths, or thousandths and this value is essential for security concerns. Depending on the cut-off point, a hacker would have more chances to break the system in the sense that a big decimal unit (tenths) produces a small synaptic weights space, which is easy to guess.

Searching the cut-off point is also an exploration of how much different decimal unit affects the accuracy of the system. In other words, this procedure shows the sensibility of the network against different noise levels inside the synaptic weights.

6.4 Performance Experiments

Two experiments on different input sizes were performed and their results were compared with the state of the results under different implementations and hardware. In this way, it is possible to get a general overview of the behavior of the proposed system with respect to the classical cryptography algorithms.

- **Experiment 1: Measuring the encryption time.** The execution time taken of the proposed system was evaluated over a group of data files. Those files have different sizes from 15KB to 1MB and are composed of alphanumeric characters. The elapsed time were measured using built-in C++ functions and compared with the encryption times reported by Patil, Narayankarb, G & M [100] ; Rihan, Khalid & Osman [101] and Mahajan & Sachdeva [102].
- **Experiment 2: Measuring the decryption time.** The elapsed time to decrypt the cipher texts generated in *Experiment 1* were measured and compared with the encryption times reported by Patil et al. [100] and Mahajan & Sachdeva [102].

6.5 Security Experiments

- **Experiment 3: Identifying the cut-off point.** To identify the cut-off point, the accuracy of the network was analyzed when floating values in the range [0.001-0.1] were randomly added and subtracted from all of the weights of the neural network. Then, the decimal unit that roughly produces 50% of accuracy was selected as the cut-off point
- **Experiment 4: Reliability of the key generator.** The overall security of the system highly relies on the capacity to produce completely different synaptic weights in every initialization of the system when a different login password is used. That is, the key generator should be able to create completely different synaptic weights even when the user login password varies in just one word. Therefore, a user password was fixed and the synaptic weights generated after arbitrarily changing one word of it were examined. For this experiment, the user password “hello” was used and the average Euclidean distance between synaptic weights was computed with respect to the password “hella”, “yello”, and “henlo”.
- **Experiment 5: Attacking with brute force approach.** Any encryption system should have a large enough key space to resist brute force attacks. Taking into account that the key for the proposed model is a seed sequence, a key space analysis based on seeds usage was performed to identify the robustness of the proposed system under brute force attack. Moreover, in the case that an attacker decides to brake the system by trial and error upon the synaptic weights of the autoencoder, probabilities calculation of the ways an attacker can choose the right weights of the neural network was carried out.

The key space analysis depends on two factors: (1) The number of seeds available in C++, and (2) The number of seeds used during the learning phase, namely, weights initialization and input data shuffling. The calculation follows the equation 6.1:

$$C = N^S, \quad (6.1)$$

where N is the number of seeds one can use in C++, and S is number of seeds used during the learning step.

Similarly, the synaptic weights space was computed follows the steps:

1. Identify the range of values in \mathbf{W}_E .
2. Identify the range of values in \mathbf{W}_D .
3. Using the cut-off point of the network calculate the cardinality of the sets composed by the possible values that \mathbf{W}_E and \mathbf{W}_D could have. Lets define this quantities E and D , respectively.
4. Compute $P = E^{72} * D^{72}$, where 72 stands for the multiplication of the rows and columns of \mathbf{W}_E and \mathbf{W}_D . Though the dimensions of both matrices are different, their multiplication is the same.

Chapter 7

Results and Discussion

The results of each one of the experiments described in the previous section are shown in this section.

7.1 Experiments

There are a total of 4 experiments in which their results are shown in the following subsections. The first two are performance experiments on different data sizes while the last two are about security concerns.

7.1.1 Experiment 1: Measuring the encryption time

The average encryption time of the proposed system was compared with the findings in [100], [102], and [101]. The average encryption time was obtained by running the proposed algorithm 10 times.

Patil et al. [100] carried out a performance comparison between DES, 3DES, AES, RSA, and Blowfish. Even though the hardware specification for that study was not mentioned, it was mentioned that the study was conducted under Java implementations. Among the considered metrics stands the encryption time when using files of sizes from 25KB to 1MB. For this experiment, the encryption time was measured in milliseconds. The results for this experiment are depicted in Table 7.1 and Figure 7.1. Although the proposed system overcomes DES, 3DES, AES, and RSA when encrypting 25KB of information in just 297 milliseconds, the proposed system stayed behind when using bulky files. For example, the encryption time of the proposed system to encrypt 1MB of information was 12062 milliseconds while the rest of the ciphers do not exceed 1400 milliseconds.

Table 7.1: Comparative at Encryption Time with Patil et al.

Input size	DES	3DES	AES	BLOWFISH	RSA	QUINGA
25KB	490	487	510	100	510	297
50KB	500	480	600	470	800	610
1MB	790	790	610	500	1400	12062

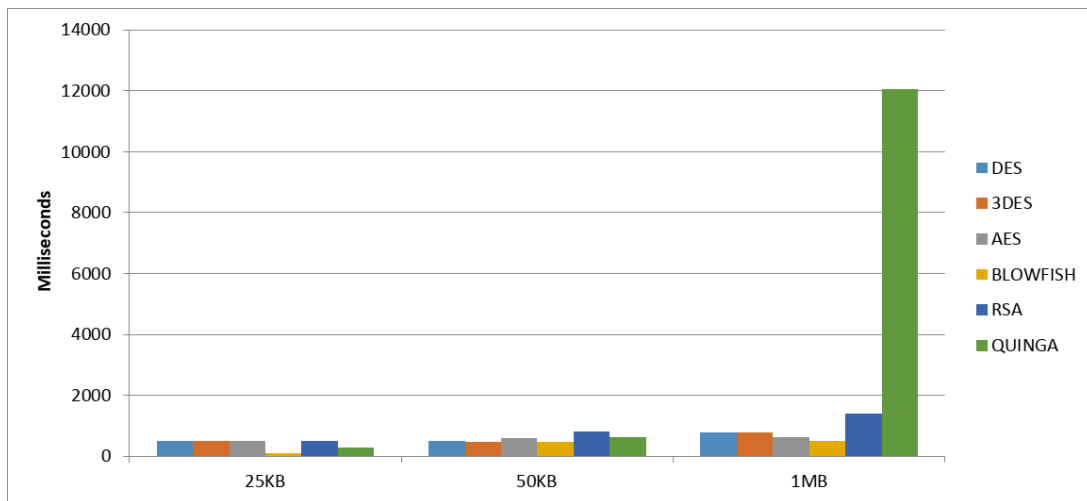


Figure 7.1: Comparative at Encryption Time with Patil et al.

A comparative study between AES and DES was conducted by Rihan et al. [101]. That experiment was performed with operating system windows 7 and a laptop core I5, 2.5 Ghz (almost the same platform used in this work). Different size text files between 15KB and 90KB were used. For this experiment, the encryption time was measured in seconds. The results obtained from this experiment are summarized in Table 7.2 and Figure 7.2. It can be observed that the proposed system tremendously overcome AES and DES regardless of the data file size. Encryption times less than one second compared with encryption times greater than 3.8 seconds shows the effectiveness of the proposed system.

Table 7.2: Comparative at Encryption Time with Rihan et al.

Input size	AES	DES	QUINGA
15KB	3.8	5.07	0.141
30KB	7.5	17.09	0.36
45KB	8.5	21.96	0.562
60KB	8.8	22.91	0.828
75KB	9.33	29.99	0.766
90KB	10.7	38.15	0.797

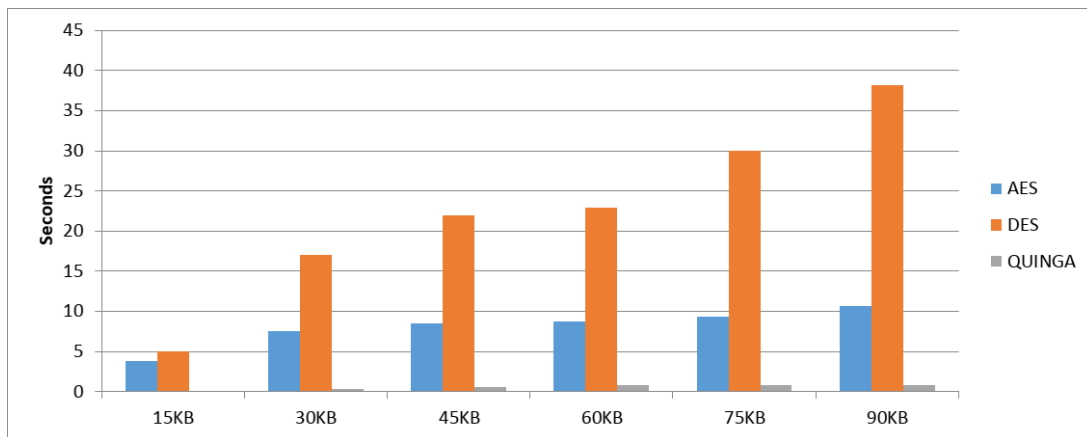


Figure 7.2: Comparative at Encryption Time with Rihan et al.

A study of encryption algorithms AES, DES, and RSA was conducted by Mahajan & Sachdeva. Text files of sizes between 153KB and 868KB were considered in this experiment and the encryption time was measured in seconds. This paper does not provide implementation details. Table 7.3 and Figure 7.3 summarize the results of this experiment. The results show that the proposed system stays within similar encryption times than AES and DES for packet sizes of 153KB, 196KB, and 312KB. However, its performance degrades when a text file of a size of 868KB is used.

Table 7.3: Comparative at Encryption Time with Mahajan & Sachdeva

Input size	AES	DES	RSA	QUINGA
153KB	1.6	3	7.3	1.7
196KB	1.7	2	8.5	2.14
312KB	1.8	3	7.8	3.53
868KB	2	4	8.2	8.39

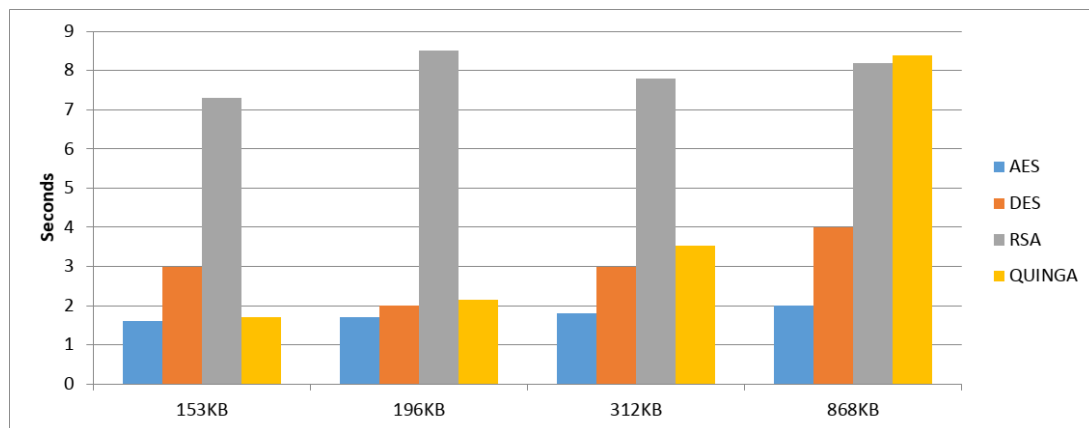


Figure 7.3: Comparative at Encryption Time with Mahajan & Sachdeva

7.1.2 Experiment 2: Measuring the decryption time

The second set of experiments was done on decryption time. For this experiment, the average decryption time of the proposed system obtained by running it 10 times as compared with the results in [100] and [102].

First, the decryption time was compared with the results obtained by Patil et al. [100]. Table 7.4 and Figure 7.4 summarize the results of this experiment. The results suggest that although all the traditional algorithms take less time for decryption than encryption, the proposed system remains almost the same. Furthermore, it can be observed that the proposed scheme takes the highest time when the file size is greater than 25KB.

Table 7.4: Comparative at Decryption Time with Patil et al.

Input size	DES	3DES	AES	BLOWFISH	RSA	QUINGA
25KB	400	390	385	50	370	297
50KB	390	390	580	100	575	625
1MB	600	570	590	300	810	11485

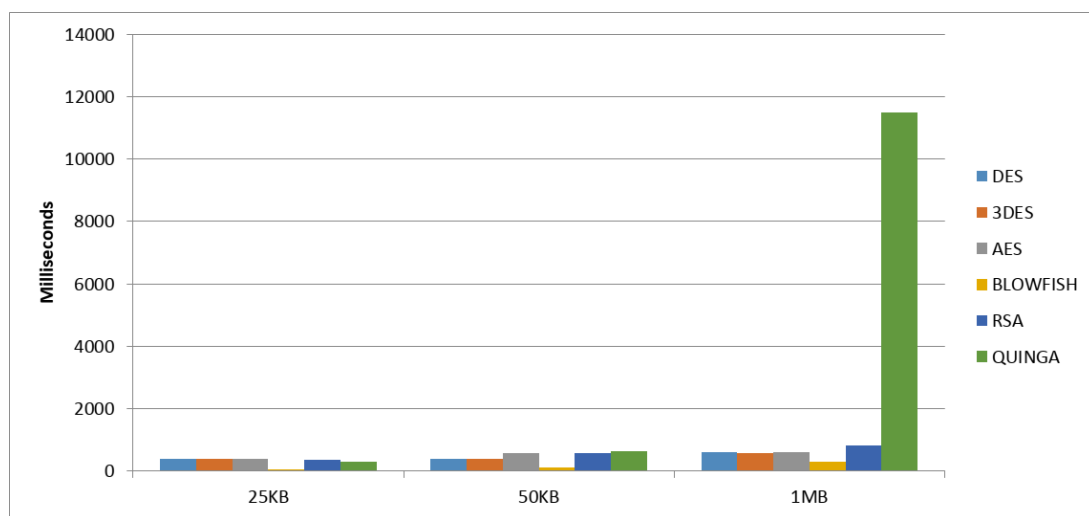


Figure 7.4: Comparative at Decryption Time with Patil et al.

Secondly, the decryption time was compared with the results obtained by Mahajan & Sachdeva [102]. Table 7.5 and Figure 7.5 summarize the results of this experiment. By analyzing the results, all the classical cipher takes less time for decryption than decryption. Nonetheless, the performance of the proposed scheme remains almost the same. It can also be observed that the proposed scheme takes less time than RSA for 153KB, 196KB, and 312KB. Likewise, the decryption time for the proposed approach is slightly larger than DES and AES. However, when a text file of size is used 868KB the proposed approach takes a much longer time compared to the time taken by AES, DES, and RSA.

Table 7.5: Comparative at Decryption Time with Mahajan & Sachdeva.

Input size	AES	DES	RSA	QUINGA
153KB	1	1.1	4.9	1.84
196KB	1.4	1.24	5.9	2.18
312KB	1.6	1.3	5.1	3.65
868KB	1.8	1.2	5.1	11.98

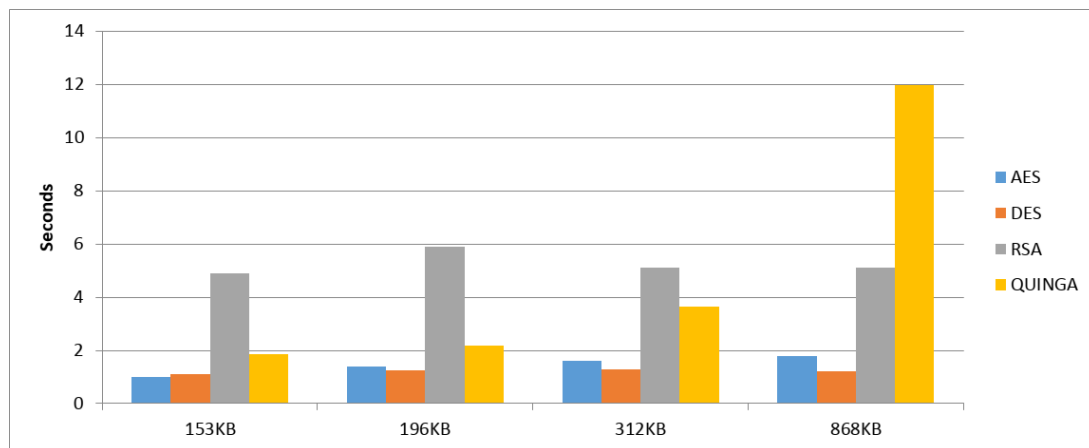


Figure 7.5: Comparative at Decryption Time with Mahajan & Sachdeva.

7.1.3 Experiment 3: Identifying the cut-off point

Different noise levels in the decimal domains of tenths, hundredths, thousandths were applied over the synaptic weights of the autoencoder to explore the sensibility of the network. The results of this experiment are shown in Table 7.6. It can be observed that the decimal unit from which 50% of accuracy was achieved are hundredths. In this sense, a depiction of the network sensibility inside the hundredths domain is shown in Figure A.2. It can also be noted that the proposed system is very sensitive to noise levels greater than 0.001. In this sense, only the first three decimals should be considered when handling the synaptic weights of the network. Besides, tenths values extremely reduce the performance of the proposed system.

Table 7.6: Network Sensibility Against Different Decimal Units, namely: Thousandths, Hundredths and Tenths.

Thousandths		Hundredths		Tenths	
Noise	Accuracy	Noise	Accuracy	Noise	Accuracy
0.001	100	0.01	99.45	0.1	36.17
0.002	100	0.02	96.17	0.2	9.76
0.003	100	0.03	89.06	0.3	3.35
0.004	100	0.04	77.73	0.4	2.18
0.005	100	0.05	75.54	0.5	0.93
0.006	100	0.06	68.04	0.6	0.85
0.007	100	0.07	53.12	0.7	0.54
0.008	99.84	0.08	58.51	0.8	0.46
0.009	99.60	0.09	42.73	0.9	0.625

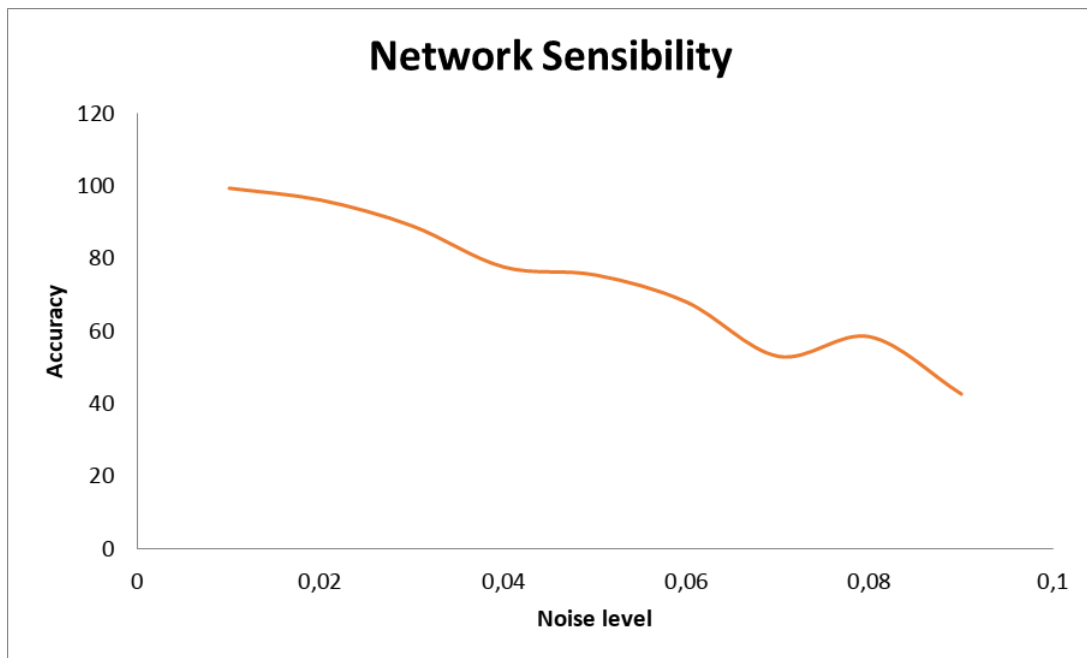


Figure 7.6: Sensibility of the network against different noise levels in the domain of the hundredths domain .

Table 7.7: Average Distance Between Synaptic with Respect to the Password "hello".

Password	Average Distance	
	W_E	W_D
"hella"	0.27	0.99
"yello"	0.15	0.66
"henlo"	0.25	0.95
Average	0.22	0.87

7.1.4 Experiment 4: Reliability of the key generator

Table 7.7. sums up the results of this experiment. It depicts the high level of variability introduced when different login password are used. It shows that the proposed key generator has the power to produce entirely different seed sequences, and therefore different synaptic weights, even with minimal changes in the login password. Form the results, it can be stated that the high level of entropy the seeds are generated with, leads to an average distance of 0.22 for W_E and 0.87 for W_D , which are values that contain tenths greater or equal than 0.1. Given that the network is very sensible to tenths, the key generator can produce synaptic weights far enough between ANN generated with different login passwords to significantly reduce the performance of the system when an encrypted message is decrypted with an autoencoder generated with a different key than the one that produced such codification. Therefore, security is increased in the sense that if an attacker gets brake the system, that effort will not affect the system when changing the login password and training again the ANN.

7.1.5 Experiment 5: Attacking with brute force approach

The cipher codes for the proposed system are: $\text{key}=\{N, S\}$, where N (number of possible seeds in C++) is 65536, S is the number of used seeds which is 144 (for weights initialization) and 14 (for input data shuffling during 14 epochs, which is the average amount of epochs required to learn the model). Hence, for this experimental task, the key space approximates to 10^{761} .

Likewise, weight codes for the proposed system are: $\text{weights}=\{E, D\}$. After several runs with different user passwords, it was identified that roughly the ranges of values for W_E and W_D are $[-0.50;0.45]$ and $[-1.74;1.69]$, respectively. Then, using the cut-off point, E is 95 and D is 393. Therefore, the weights space approximates to $8, 56 * 10^{324}$.

By the proposed key and weights space analysis, that leads to quantities of the order of 10^{761} and 10^{324} , the proposed system emerges as a extremely hard to crack through a brute-force effort.

Chapter 8

Conclusions and Perspectives

The main aim of this work was to propose a new symmetric key cryptography scheme to overcome the speed and security faults of traditional ciphers. Based on a state of the art analysis, we identified that most of the common cipher algorithms used today, namely 3DES, AES and RSA are based on complex serial computations that slow down their performance, as well as, their security vulnerabilities have been exposed. Therefore, we propose a new deep learning-based approach using neural networks which represents a way of the next development in cryptography. We experiment with the performance of the proposed system with real-world text files with different sizes. Furthermore, experimental tasks were performed to evaluate the robustness of the system against a brute force attack. We conclude that the effectiveness and reliability of the proposed cryptographic system are quite compelling. In a more detailed fashion the following conclusions are pointed out:

1. The encryption and decryption times were compared with the state of the art findings to generalize results. Experimental results showed that the proposed system tremendously overcome DES, 3DES, AES, and RSA when encrypting text data files of sizes between 15KB and 90KB. In addition, for files of sizes between 153KB and 312KB the performance of the proposed system is comparable to the performance of the traditional ones. On the contrary, the performance of the proposed system considerably decreases for data files of sizes greater than 868KB.
2. In terms of the noise-accuracy ratio of the autoencoder, it can be noted that the accuracy starts to decrease, from 100%, when a noise level greater than 0.008 affects the synaptic weights of the ANN. In this sense, only the first three decimals should be considered when handling the synaptic weights of the network. Similarly, the value 0.07 roughly reduces the accuracy of the autoencoder to 50%, and therefore hundredths decimal units were considered in the security analysis of the proposed cryptosystem. Moreover, tenths values extremely reduce the accuracy of the proposed system.
3. Considering that the network is very sensible to tenths, the key generator can produce synaptic weights far enough between them (0.22 and 0.87) to make impossible the decodification of a cipher text with an autoencoder generated with a different key than the one that produced such codification. Therefore, security is increased in the sense that if an

attacker gets brake the system, that effort will not affect another instance of the network, which is produced using a different password.

4. The security analysis of the proposed system is reduced to identify the robustness of the system against brute force efforts. Every encryption algorithm should have a large enough key space to resits brute force attacks. By the proposed key and weights space analysis, that leads to quantities of the order of 10^{761} and 10^{324} respectively, the proposed system emerges as an extremely hard to crack through a brute-force effort. Especially, when it is compared with the order of magnitudes attacks for classical ciphers require

As future work, since cryptography is used not only for text files but also for other multimedia formats, we will work to modify the proposed scheme and make it able to utilize image and audio data. Apart from that, as it was discussed in this work, there are various attacks. Each attack uses different techniques and can take different amounts of time to break the system. Furthermore, each attack exploits known information about the encryption scheme. Therefore, an exploration of different methods for cracking the proposed system will be performed.

Moreover, since there are multiple programming languages specialized for artificial intelligence applications, an exploration of the performance of the proposed system under different implementation is going to be carried out.

References

- [1] S. Vaudenay, *A CLASSICAL INTRODUCTION TO CRYPTOGRAPHY Applications for Communications Security*. Springer, 2006.
- [2] S. Singh, *The Code Book, How to make it, break it, hack it, crack it*. Delacorte Press, 2001.
- [3] M. Franchi, “Algoritmos de encriptación de clave asimétrica,” Oct. 2012.
- [4] Y. Medina and H. Miranda, “Comparison of algorithms based cryptography symmetric des, aes and 3des,” *MUNDO FESC*, pp. 14–21, Jun. 2015.
- [5] S. Steffen, “Cryptographic Hash Functions,” 2009.
- [6] I. Basheer and M. Hajmeer, “Artificial neural networks: fundamentals, computing, design, and application,” *Journal of Microbiological Methods*, pp. 3–31, 2000.
- [7] 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. [Online]. Available: www.ijcsmc.com
- [8] M. Barakat, C. Eder, and T. Hanke, “An introduction to cryptography,” *Information Security Management Handbook, Sixth Edition*, pp. 1121–1139, 2018.
- [9] E. M. Nigm, O. S. Faragallah, and A. Mousa, “Cryptography and Database Security : Concepts , Compliance Risks , and Technical Challenges,” pp. 1–14.
- [10] M. Ubaidullah and Q. Makki, “A Review on Symmetric Key Encryption Techniques in Cryptography,” *International Journal of Computer Applications*, vol. 147, no. 10, pp. 43–48, 2016.
- [11] S. B, “Applied Cryptography,” *Electrical Engineering*, vol. 1, no. 32, pp. 429–455, 1996.
- [12] Gulzar and Nikki, “Surveillance Privacy Protection,” *Intelligent Multimedia Surveillance*, no. 32, pp. 83–105, 2013.
- [13] Wang and Cong, “Toward secure and dependable storage services in cloud computing,” *Services Computing*, vol. 1, no. 32, pp. 220–232, 2012.

- [14] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques," *Int. J. Comput. Sci. Eng.*, vol. 4, 05 2012.
- [15] N. Arora and Y. Gigras, "Block and Stream Cipher Based Cryptographic Algorithms: A Survey," *International Journal of Information and Computation Technology*, vol. 4, no. 2, pp. 189–196, 2014.
- [16] D. Boneh and V. Shoup, "A Graduate Course in Applied Cryptography," *Crypto.Stanford.Edu*, no. 1, pp. 1–400, 2015. [Online]. Available: <https://crypto.stanford.edu/~dabo/cryptobook/draft{-}0{-}2.pdf>
- [17] A. Vijaya Kumar, R. Maria, and S. K. Sukumar, "Kalam10 Stream Cipher," *SSRN Electronic Journal*, no. September, 2018.
- [18] P. v. O. Alfred Menezes and S. Vanstone, *Handbook of Applied Cryptograph.* CSC Press, 2001, vol. 1.
- [19] G. Singh and Supriya, "A study of encryption algorithms (rsa, des, 3des and aes) for information security," *International Journal of Computer Applications*, 2013.
- [20] X. Zhou and X. Tang, "Research and implementation of rsa algorithm for encryption and decryption," 2011.
- [21] O. M. David Kravitz, "Digital signature algorithm," Jul. 1993.
- [22] T. Shankar and G. Sahoo, "Cryptography with elliptic curves," *International Journal Of Computer Science And Applications*, May 2009.
- [23] E. Swathi, G. Vivek, and G. S. Rani, "Role of Hash Function in Cryptography," no. November, pp. 10–13, 2016.
- [24] R. Tripathi and S. Agrawal, "Comparative Study of Symmetric and Asymmetric Cryptography," *International Journal of Advance Foundation and Research in Computer*, vol. 1, no. 6, pp. 68 – 76, 2014. [Online]. Available: <https://pdfs.semanticscholar.org/e0e4/810c5276f9c05cc82425fcf911f206c52bef.pdf>
- [25] S. Iqbal, "Symmetric Key Cryptography : Technological Developments in the Field," vol. 117, no. 15, pp. 23–26, 2015.
- [26] S. William, *Cryptography and Network Security.* Prentice Hall, 2005, vol. 1025, no. 1.
- [27] C. Greg, *Security data visualization: graphical techniques for network analysis.* Starch Press, 2005, no. 1.
- [28] Y. Kumar, R. Munjal, and H. Sharma, "Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures," *IJCSMS International Journal of Computer Science and Management Studies*, vol. 11, no. 03, pp. 2231–5268, 2011. [Online]. Available: www.ijcsms.com

- [29] A. Wahid, A. A. Esparham, and B. Marwan, "A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention," *Journal of Computer Science Applications and Information Technology*, vol. 3, no. 2, pp. 1–8, 2018. [Online]. Available: <https://symbiosisonlinepublishing.com/computer-science-technology/computerscience-information-technology32.php>
- [30] M. M. Alani, "Neuro-cryptanalysis of des and triple-DES," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7667 LNCS, no. PART 5, pp. 637–646, 2012.
- [31] S. Lucks, "Attacking triple encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1372, pp. 239–253, 1998.
- [32] A. Biryukov and D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5912 LNCS, pp. 1–18, 2009.
- [33] D. Khovratovich, *Biclique Cryptanalysis of the Full AES Biclique Cryptanalysis of the Full AES*, 2014, no. July.
- [34] C. F. Cid, "Cryptanalysis of RSA: A Survey," *Sans*, 2019.
- [35] B. K and D. S.S, "An Overview of Cryptanalysis of RSA Public key System," *International Journal of Engineering and Technology*, vol. 9, no. 5, pp. 3575–3579, 2017.
- [36] L.Deng and D.Yu, "Deep Learning: Methods and Applications," *now Publishers Inc.*, vol. 7, no. 2, pp. 197–387, 2014.
- [37] Y.Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
- [38] A. Begum, F. Fatima, and A. Sabahath, "Implementation of Deep Learning Algorithm with Perceptron using TensorFlow Library," *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 172–175, 2019.
- [39] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [40] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning."
- [41] J. Zurada, *Introduction to Artificial Neural System*. PWD, 1992.
- [42] D. M. Joshi and N. K. Rana, "Classification of Brain Cancer Using Artificial Neural Network," *2010 2nd International Conference on Electronic Computer Technology*, pp. 112–116, 2010.

- [43] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research,” *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.
- [44] I. A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application,” *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000.
- [45] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of trends in Practice and Research for Deep Learning,” pp. 1–20, 2018. [Online]. Available: <http://arxiv.org/abs/1811.03378>
- [46] “Types of Machine Learning Algorithms,” *New Advances in Machine Learning*, no. February 2010, 2010.
- [47] M. Mohammed, M. B. Khan, and E. B. M. Bashie, *Machine learning: Algorithms and applications*, 2016, no. December.
- [48] R. Sathya and A. Abraham, “Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.
- [49] S. Zhang, C. Zhang, and Q. Yang, “Data preparation for data mining,” *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 375–381, 2003.
- [50] L. Yu and H. Liu, “Efficient feature selection via analysis of relevance and redundancy,” *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.
- [51] O. and F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, “Supervised Machine Learning Algorithms: Classification and Comparison,” *International Journal of Computer Trends and Technology*, vol. 48, no. 3, pp. 128–138, 2017.
- [52] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, “Engineering applications of the self-organizing map,” *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1383, 1996.
- [53] Z. Ghahramani, “Unsupervised learning algorithms are designed to extract structure from data,” *IOS ppress*, vol. 178, no. 10, pp. 1–8, 2008.
- [54] G. Zoubin, “Unsupervised Learning ,” pp. 1–32, 2004.
- [55] O. Chapelle and B. Schölkopf, *Semi-Supervised Learning*.
- [56] M. Seeger, “Learning with labeled and unlabeled data,” Institute for Adaptive and Neural Computation, Edinburgh, Tech. Rep., 2001.
- [57] X. Zhu, “Semi-Supervised Learning,” pp. 1–10.
- [58] R. S. Sutton and A. G. Barto, “Reinforcement Learning : An Introduction,” 2015.

- [59] C. Szepesvári, “Algorithms for reinforcement learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 9, pp. 1–89, 2010.
- [60] M. E. Harmon, W. . Aacf, and S. S. Harmon, “Reinforcement Learning: A Tutorial Scope of Tutorial.”
- [61] I. N. da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, “Artificial neural networks: A practical course,” *Artificial Neural Networks: A Practical Course*, pp. 1–307, 2016.
- [62] M. T. Hagan, H. B. Demouth, M. H. Beale, and O. De Jesús, *Neural Network Design*, 2015, vol. 12, no. 3.
- [63] B. M. Wilamowski, “Neural network architectures,” *Intelligent Systems*, pp. 1–18, 2016.
- [64] B. Widrow and A. Micahel, “Artificial Neural Netwroks of The Perceptron Madaline and Backpropagation,” *Neurobonics*, 1992.
- [65] J. A. Bullinaria, “12- Recurrent Neural Networks,” *Neural Computation : Lecture 12*, pp. 1–20, 2015. [Online]. Available: <https://www.cs.bham.ac.uk/~jxb/inc.html>
- [66] S. Haykin, *Neural Networks. A Comprehensive Foundation*, 2nd ed. Hamilton, Ontario, Canada: Pearson Education, 2005.
- [67] R. Rojas, “Neural networks: A systematic introduction,” 1996.
- [68] N. Buduma, “Fundamentals of deep learning: Designing next-generation artificial intelligence algorithms,” 2015.
- [69] A. Ng, “Sparse Autoencoder,” ”*CS294A Lecture Notes*, vol. 72, pp. 1–19, 2011.
- [70] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1,” D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362. [Online]. Available: <http://dl.acm.org/citation.cfm?id=104279.104293>
- [71] P. Baldi, “Boolean AutoEncoder,” pp. 37–50, 2012. [Online]. Available: <http://proceedings.mlr.press/v27/baldi12a/baldi12a.pdf>
- [72] G. Hinton and R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” vol. 313, no. July, pp. 504–507, 2006.
- [73] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, and P. Vincent, “Why Does Unsupervised Pre-training Help Deep Learning?” *Journal of Machine Learning Research*, vol. 11, no. 20, pp. 625–660, 2010.
- [74] R. M. Jogdand and S. S. Bisalapur, “Design of An Efficient Neural Key Generation,” *International Journal of Artificial Intelligence & Applications*, vol. 2, no. 1, pp. 60–69, 2011.

- [75] W. Kinzel and I. Kanter, "Neural cryptography," *ICONIP 2002 - Proceedings of the 9th International Conference on Neural Information Processing: Computational Intelligence for the E-Age*, vol. 3, no. November, pp. 1351–1354, 2002.
- [76] A. Singh and A. Nandal, "Neural Cryptography for Secret Key Exchange and Encryption with AES," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, pp. 376–381, 2013.
- [77] 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.
- [78] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2501, pp. 288–298, 2002.
- [79] E. Volna, M. Kotyrba, V. Kocian, and M. Janosek, "Cryptography based on neural network," *Proceedings - 26th European Conference on Modelling and Simulation, ECMS 2012*, no. February 2015, 2012.
- [80] P. P. Hadke and S. G. Kale, "Use of Neural Networks in cryptography: A review," *IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*, pp. 1–4, 2016.
- [81] C. K. Chan and L. M. Cheng, "Pseudorandom generator based on clipped Hopfield neural network," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 183–186, 1998.
- [82] D. A. Karras and V. Zorkadis, "Improving Pseudorandom Bit Sequence Generation and Evaluation for Secure Internet Communications Using Neural Network Techniques," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 1367–1372, 2003.
- [83] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking Cryptographic Implementations Using Deep Learning Techniques," *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10076 LNCS, pp. 3–26, 2016.
- [84] K. Jayachandiran, "A Machine Learning Approach for Cryptanalysis," pp. 81–87, 2016.
- [85] A. Gohr, "Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning," pp. 150–179, 2019.
- [86] L. Shaohui, Y. Hongxun, and G. Wen, "NEURAL NETWORK BASED STEGANALYSIS IN STILL IMAGES," pp. 509–512, 2003.

- [87] 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," *IEEE International Conference on Multimedia and Expo, ICME 2005*, vol. 2005, no. May 2014, pp. 269–272, 2005.
- [88] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," *Media Watermarking, Security, and Forensics 2015*, vol. 9409, p. 94090J, 2015.
- [89] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-Janua, no. August, pp. 1–6, 2018.
- [90] J. Ye, J. Ni, and Y. Yi, "Deep Learning Hierarchical Representations for Image Steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.
- [91] F. Hu, J. Wang, X. Xu, C. Pu, and P. Tao, "Batch Image Encryption Using Generated Deep Features Based on Stacked Autoencoder Network," *Mathematical Problems in Engineering*, vol. 2017, pp. 256–261, 2017.
- [92] S. Li, G. Chen, and X. Zheng, *Chaos-Based Encryption for Digital Images and Videos*, 12 2004.
- [93] N. Mohananthini and G. Yamuna, "A Study in Image Watermarking Schemes using Neural Networks," no. 9, 2016.
- [94] M. Mondal and K. Ray, "Review on DNA Cryptography," pp. 1–31, 2019.
- [95] M. Popli and G. ., "DNA Cryptography: A Novel Approach for Data Security Using Flower Pollination Algorithm," *SSRN Electronic Journal*, 2019.
- [96] A. Omer, "Dna cryptography," 07 2015.
- [97] S. Kalsi, H. Kaur, and V. Chang, "DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation," *Journal of Medical Systems*, vol. 42, no. 1, 2018.
- [98] S. Basu, M. Karuppiah, M. Nasipuri, A. K. Halder, and N. Radhakrishnan, "Bio-Inspired Cryptosystem with DNA Cryptography and Neural Networks," *Journal of Systems Architecture*, 2019. [Online]. Available: <https://doi.org/10.1016/j.sysarc.2019.02.005>
- [99] G. Maddodi, A. Awad, D. Awad, M. Awad, and B. Lee, "A new image encryption algorithm based on heterogeneous chaotic neural network generator and dna encoding," *Multimedia Tools and Applications*, vol. 77, no. 19, pp. 24 701–24 725, 2018.

- [100] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, “A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish,” *Procedia Computer Science*, vol. 78, no. December 2015, pp. 617–624, 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.procs.2016.02.108>
- [101] K. Ahmed and S. D. Rihan, “A Performance Comparison of Encryption Algorithms AES and DES,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, no. December, pp. 151–154, 2015. [Online]. Available: <https://www.ijert.org/research/a-performance-comparison-of-encryption-algorithms-aes-and-des-IJERTV4IS120227.pdf>
- [102] P. Mahajan and A. Sachdeva, “A Study of Encryption Algorithms AES, DES and RSA for Security,” *Global Journal of Computer Science and Technology Network, Web & Security*, vol. 13, no. 15, pp. 64–69, 2013.

Appendices

Appendix A

Cryptosystem Prototype

In this Section, a GUI for monitoring the neural network while training it, as well as, a short message encryption and decryption demonstration are depicted.

A.1 GUI for monitoring the neural network training

During the first part of this degree work, namely the design and calibration of an autoencoder for encryption and decryption of the complete ASCII set, a GUI was designed for monitoring the neural network while it was training.

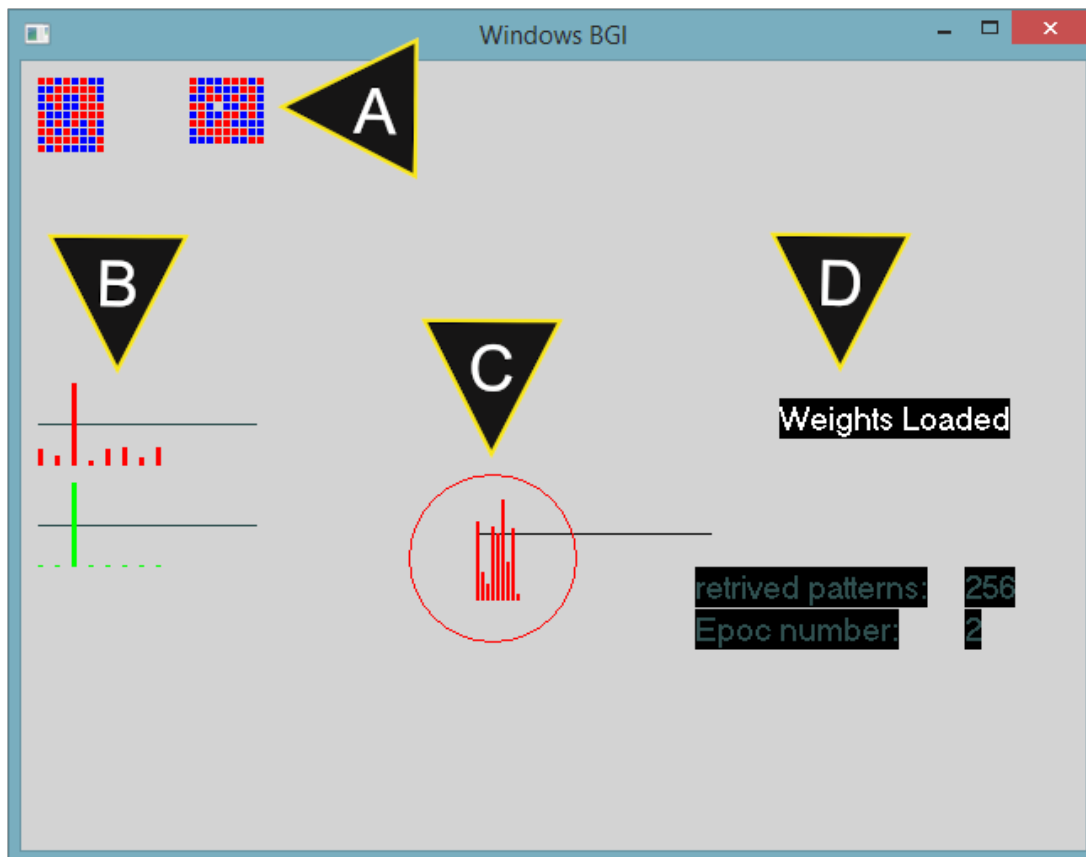
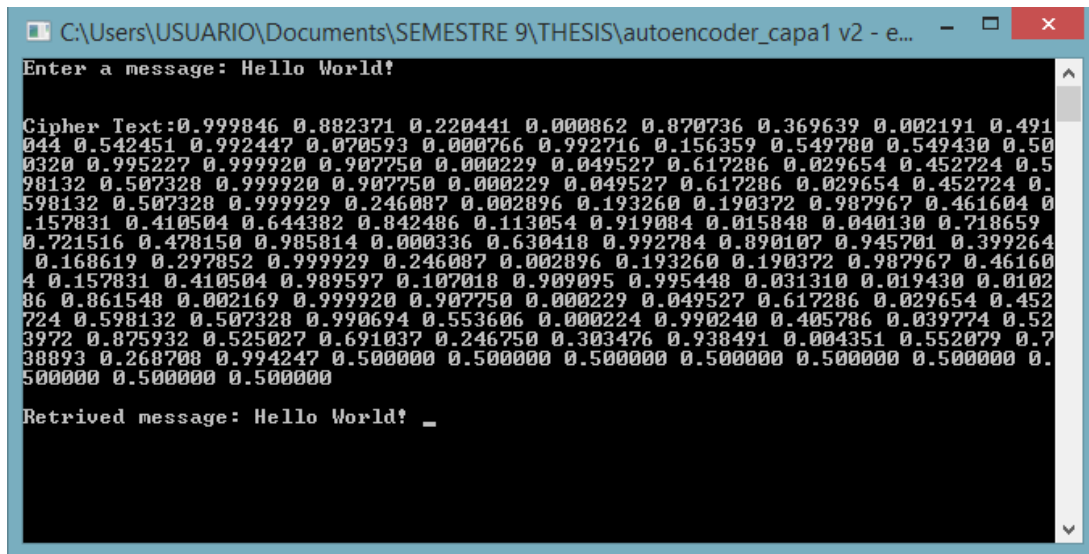


Figure A.1: GUI for monitoring the autoencoder training. (A): W_E and W_D representation, the red signal means excitatory response while the blue one means an inhibitory response, (B) Expected and actual values, the former is represented by red lines and the latter by green lines, (C) Hidden values representation, i.e the cipher code, (D) Reports about the number of retrieved patterns and elapsed epochs.

A.2 Encryption and decryption demonstration

Although several text files of different sizes were successfully encrypted and decrypted, in this section a short message is shown for simplifying the depiction of the system at work.



```
C:\Users\USUARIO\Documents\SEMESTRE 9\THESIS\autoencoder_capa1 v2 - e... - □ ×
Enter a message: Hello World!

Cipher Text:0.999846 0.882371 0.220441 0.000862 0.870736 0.369639 0.002191 0.491
044 0.542451 0.992447 0.070593 0.000766 0.992716 0.156359 0.549780 0.549430 0.50
0320 0.995227 0.999920 0.907750 0.000229 0.049527 0.617286 0.029654 0.452724 0.5
98132 0.507328 0.999920 0.907750 0.000229 0.049527 0.617286 0.029654 0.452724 0.
598132 0.507328 0.999929 0.246087 0.002896 0.193260 0.190372 0.987967 0.461604 0.
157831 0.410504 0.644382 0.842486 0.113054 0.919084 0.015848 0.040130 0.718659
0.721516 0.478150 0.985814 0.000336 0.630418 0.992784 0.890107 0.945701 0.399264
0.168619 0.297852 0.999929 0.246087 0.002896 0.193260 0.190372 0.987967 0.46160
4 0.157831 0.410504 0.989597 0.107018 0.909095 0.995448 0.031310 0.019430 0.0102
86 0.861548 0.002169 0.999920 0.907750 0.000229 0.049527 0.617286 0.029654 0.452
724 0.598132 0.507328 0.990694 0.553606 0.000224 0.990240 0.405786 0.039774 0.52
3972 0.875932 0.525027 0.691037 0.246750 0.303476 0.938491 0.004351 0.552079 0.7
38893 0.268708 0.994247 0.500000 0.500000 0.500000 0.500000 0.500000 0.500000 0.
500000 0.500000 0.500000

Retrieved message: Hello World! _
```

Figure A.2: Depiction of the functionality of the proposed system.