Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Sciences for Women
Department of Computer Sciences

# Implementation of Message Authentication Code Algorithm in Multi-core Environment

**A Thesis**

Submitted to the Council of College of Sciences for Women, University of Babylon in a Partial Fulfilment of the Requirements for the Degree of Master in Science\ Computer Sciences

**By**

**Yamamah Alaa Abdulkadhim**

**Supervised By**

***Asst.Prof.Dr.*** ***Asst.Prof.Dr.***

**Ahmed Badri Muslim** **Esraa Hadi Abaid**

**2023 A. D.** **1445. A.H.**

بسم الله الرحمن الرحيم

﴿نَرْفَعُ دَرَجَاتٍ مَنْ نَشَاءُ ۗ

وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴾

صَدَقَ ٱللَّهُ ٱلْعَظِيمُ

يوسف﴿ 76 ﴾

## CERTIFICATION OF THE EXAMINATION COMMITTEE

We are the members of the examination committee, certify that we have read this thesis entitled (**Implementation of Message Authentication Code Algorithm in Multi-core Environment**) and after examining the master student (**Yamamah Alaa Abdulkadhim**) in its contents on 12/10/2023 and that in our opinion it is adequate as a thesis for the degree of Master in Science /Computer Science with degree (Excellence)

**Committee Chairman**

Signature:
Name: **Wesam Samir abd ali**
Scientific Degree: **Prof. Dr.**
Address: University of Babylon / College of Information Technology
Date: / /2023

**Committee Member**

Signature:
Name: **Ali Shakir Mahmood**
Scientific Degree: **Asst. Prof. Dr.**
Address: University of Al mustansiriyah/ College of Education Computer Science Department
Date: / / 2023

**Committee Member**

Signature:
Name: **Saif Al-alak**
Scientific Degree: **Asst. Prof. Dr.**
Address: University of Babylon/ College of Science for Woman
Date: / / 2023

**Supervisor**

Signature:
Name: **Ahmad Badri Muslim**
Scientific Degree: **Asst.Prof. Dr.**
Address: University of Babylon/ College of Science for Woman
Date: / / 2023

**Supervisor**

Signature:
Name: **Esraa Hadi Abaid**
Scientific Degree: **Asst.Prof. Dr.**
Address: University of Babylon/ College of Science for Woman
Date: / / 2023

Date of Examination: 12/10/2023
Deanship Authentication of College of Science for Woman
Approved for the College Committee of graduate studies.

Signature:
Name: **Abeer Fauzi Al-Rubaye**
Scientific Degree: **Prof. Dr.**
Address: **Dean of College of Science for Woman**
Date: / / 2023

## *Supervisors Certification*

***We certify that this thesis entitled "Implementation of Message Authentication Code Algorithm in Multi-core Environment" is done by (Yamamah Alaa Abdualkathm) under our supervision.***

***Signature:***
***Name: Asst.Prof.Dr. Ahmed Badri Muslim***
***Date: / /2023***
***Address: University of Babylon/College of Science for Women***

***Signature:***

***Name: Asst.Prof. Dr. Esraa Hadi Abaid***

***Date: / / 2023***

***Address: University of Babylon/College of Science for Women***

## *Head of the Department Certification*

***In view of the available recommendations, I forward the thesis entitled "Implementation of Message Authentication Code Algorithm in Multi-core Environment" for debate by the examining committee.***

***Signature:***
***Name: Asst. Prof. Dr. Saif Al-alak***
***Date: / / 2023***
***Address: University of Babylon/College of Science for Women***

# *Dedication*

*I dedicate this thesis*

*To whom their presence in my life pushed me to provide the best...to the path of my success and the light of my success, my heaven on earth (my mother and my father)*

*To whom I had the best help and support throughout my studies (my dear husband)*

*To my strength in my weakness (my dear brothers)*

*To those whom I see as a light illuminating my heart...To those who made me look at life with the spirit of an innocent child, at the fragrance of winds, the butterflies of orchards, and the fruits of life, my beautiful princesses (my daughters)*

*To my shadow and my mirror, to my happiness, and to the source of my laughter and joy (my dear friends).*

**Yamamah**

## Acknowledgments

In the name of Allah most gracious and merciful. I am thankful to my creator who blessed me with the ability to complete this thesis and for everything.

I would like to express my deepest appreciation and gratitude to my supervisors **Asst. Prof. Dr. *Ahmed Badri Muslim*** and **Asst. Prof. Dr. *Esraa Hadi Abaid*** for their guidance, valuable advice, helpful discussions, thoughtful comments, continuous encouragement and support throughout the year of study and research at every step of this thesis.

# Abstract

There are around 4.95 billion internet users worldwide at present, making this a sizable market with steadily rising demand for online services. These include things like internet socializing and information sharing as well as online purchasing. This highlights the critical requirement for more privacy and secrecy. Online fraud has emerged as one of the most significant obstacles to the widespread use of business apps. This has led to the emergence of concerns about authentication, authorization, and identification as important challenges in our modern, open society. The identification procedure is the act of recognizing a person, computer, or software application. Security systems utilize authentication and authorization together to determine who is allowed to access data and systems via a network. The user's gadget is responsible for authentication, while the security system is in charge of authorization. Parallelism, however, has been proposed as a means to increase the effectiveness of authentication procedures among other potential alternatives. This thesis, is set out to create a brand-new algorithm for authenticating messages in parallel across several cores of a computer. Two PNGR files and two surrogate boxes are used in the proposed message authentication procedure to encode and verify the regular message. The cipher only requires one round to encrypt or decrypt a block of data, making it a very efficient method. When implemented on a multi-core CPU, the suggested authentication method is 3.27 times faster than the spot-based parallel authentication method. When comparing the suggested algorithm of parallel implementation to its serial counterpart, an average speedup of 2.99 is achieved. The proposed method is shown to be sufficiently random to pass the most stringent test, and the resulting MAC values are also shown to pass a most difficult test of security.

# List of Contents

## List of Tables

## List of Figures

## List of Abbreviations

| Abbreviations | Meaning |
|---|---|
| **AES** | Advanced Encryption Standard |
| **ILP** | Instruction-Level Parallelism |
| **KSA** | Key Setup Algorithm |
| **LEA** | Low-Power Encryption Algorithm |
| **MD5** | Message Digest Algorithm 5 |
| **MPI** | Message-Passing Interface |
| **MAC** | Message Authentication Code |
| **PKI** | Public Key Infrastructure |
| **RC4** | Rivest Cipher 4 |
| **RSA** | Rivest-Shamir-Adleman |
| **SHA-1** | Secure Hash Algorithm 1 |
| **SHA-256** | Secure Hash Algorithm 256 |
| **SHA-3** | Secure Hash Algorithm 3 |
| **TALP** | Task-Level Parallelism |
| **TLP** | Thread-Level Parallelism |
| **3DES** | Triple Data Encryption Standard |

# CHAPTER ONE

# GENERAL INTRODUCTION

## 1.1 Introduction

In today's data-driven and interconnected world, ensuring the authenticity and integrity of messages is of paramount importance. Message authentication algorithms play a crucial role in verifying the integrity of transmitted data, detecting any unauthorized modifications or tampering. Computer systems have grown rapidly during the last 20 years due to the development of networks, which have become vital tools in a variety of environments. Moreover, networks are indeed being developed by companies in greater proportions than ever before, and access to the global Internet has become essential [1]. This pattern has been matched by an increase in unauthorized access to computer systems through the use of computer networks. Everyone knows that the Internet is a strong platform that transforms how we do business in today's technology. It now influences every facet of our lives, in addition to the emergence of more security-related dangers [2].

In recent years, Internet use has grown at an accelerating rate and has become an integral aspect of daily life. It is believed that there are now more than 4.95 billion users of the internet worldwide [3]. Therefore, the appearance of the information security has significantly changed our lives, especially with the availability of information and the flexibility to view and modify the data. With the advent of Internet banking and electronic commercial exchanges, activities that were previously primarily conducted offline, like banking and market trades, are now primarily conducted online [2]. However, protection against threats to many other types of security services, such as source authentication, data integrity, and confidentiality, is essential. The safety of these services is typically guaranteed by using cryptographic techniques. There are now two categories of assaults, active and passive. Both active and passive assaults pose serious threats to data confidentiality and user privacy, although active attacks are more likely to compromise data authenticity, integrity, and availability. Furthermore, a proactive attacker may add, change, or delete information at will. By passively reading or copying the contents of a communication, an attacker might compromise its secrecy [4]. The passive assault has no effect on the system. One of the most important characteristics of a passive assault is that the target is usually unaware of it [5]. Active and passive assaults are related. Such attacks will do a lot of damage. The proliferation of both data and the Internet means that it's more important than ever to take precautions against potential threats to any and all data, regardless of its format, by ensuring its confidentiality, integrity, and authenticity. The problems caused by message authentication attacks may be circumvented by the use of message authentication algorithms (MAA). The basic goal of any technique for authenticating messages is to guarantee that only those with knowledge of the same secret key can read and understand the message during transmission. This prevents an adversary from manipulating the message in a way that might have unintended consequences. Symmetric message authentication often employs block cipher-based or hash-based approaches. [5]. Separating the message into chunks, each of which has its own dynamic key, is the method proposed. Each thread then independently implements the message authentication procedure for the local MAC value, and finally all threads collaborate on the global MAC account.

## 1.2 Problem of Statement

Traditional message authentication algorithms may struggle to meet the performance requirements imposed by the increasing volume of messages and the demand for real-time authentication. The sequential algorithms process messages one at a time, leading to potential bottlenecks and high latency in the authentication process. As a result, the authentication speed may not be sufficient to handle large-scale message authentication tasks in time-sensitive applications. The problem lies in finding an efficient and scalable solution that can expedite the message authentication process without compromising security or accuracy. This solution should be able to take advantage of the parallel processing capabilities offered by modern multi-core CPUs, which can execute multiple tasks simultaneously.

The main challenges to address in the parallel message authentication method are:

1. Performance: The authentication method should significantly improve the performance of message authentication by leveraging parallel processing. It should distribute the authentication tasks across multiple CPU cores to achieve faster authentication times and increase overall system throughput.
2. Latency: The authentication method should reduce the authentication latency, enabling real-time authentication in time-sensitive applications. By processing messages in parallel, the method should minimize the time required to authenticate a batch of messages and provide low-latency authentication responses.
3. Security: While improving performance, the parallel message authentication method must ensure that the security and integrity of the authentication process are not compromised. It should maintain the same level of security as traditional sequential algorithms, preventing unauthorized modifications or tampering of messages.

Addressing these challenges will lead to a parallel message authentication method that offers improved performance, scalability, reduced latency, and efficient resource utilization, making it suitable for high-performance message authentication in various applications.

## 1.3 Thesis of Aims

The objective of this thesis is to propose and implement a parallel message authentication algorithm over a multicore CPU to enhance the efficiency and speed of authentication methods. The research aims to address the increasing demand for online activities while improving levels of secrecy and privacy. The specific objectives include:

1-Developing a new message authentication algorithm that utilizes parallelism and multicore processing for improved performance.

2-Conducting rigorous testing and analysis to ensure randomness and security of the generated MAC values.

3-Assessing the performance of the proposed algorithm in terms of speedup and efficiency compared to the sequential version of the algorithm. Providing recommendations for the practical implementation and deployment of the proposed algorithm in real-world applications.

## 1.4 Thesis of Contribution

The Parallel Message Authentication Algorithm (PMAA) implemented over a multi-core CPU offers several main contributions:

- Enhanced Throughput: The parallel execution of PMAA on multi-core CPUs improves the throughput of message authentication. With multiple cores working in parallel, the algorithm can authenticate a larger number of messages simultaneously. This increased throughput is particularly beneficial in high-load scenarios, such as network communication or data processing applications, where numerous messages require authentication within a limited timeframe.

Overall, the main contributions of implementing PMAA over a multi-core CPU include improved performance, scalability, reduced latency, enhanced throughput, and efficient resource utilization, making it a suitable choice for high-performance message authentication in various applications.

## 1.5 Related Work

Communication security necessitates the use of Secure-hash algorithms (SHAs), with SHA-256 being the most secure and efficient option. Most signature algorithms, including quantum-resistant ones, utilize SHA-256 for this purpose. A recent work presented in [6] proposes a concurrent SHA-256 implementation for hashing hundreds of messages on the Sunway TaihuLight SW26010 many-core processor, one of the world's fastest and most powerful computers.

The Karatsuba-Ofman Algorithm (KOA) is the name of the algorithm used for multiplying Galois Fields (GF). Unlike previous parallel hardware KOA-based systems, the authors of [7] utilized a single reduction array for all parallel multipliers, which resulted in an optimized design in terms of area utilization.

In [5], it has been suggested that the new Message Authentication Algorithm (MAA), which will be known as 'DKEMA', would better accommodate the capabilities and notions of the GPU. The dynamic key-dependent scheme with one round of substitution and diffusion operations forms the basis for this system. The results of the experiments indicate that the suggested approach is quite successful on a Titan V100 GPU; the throughput is more than 400 GB/s.

In a recent publication [8], a new authentication mechanism called parallel cipher-based message authentication code (PCMAC) was proposed by researchers. The method was designed in a parallel structure, and it is highly efficient for applications that demand high throughput. A comparative analysis was conducted between the suggested technique and MAC-based authentication schemes. The findings of the study demonstrated that the PCMAC method can authenticate data at a speed of 2.99 GB/s.

In [9], researchers present a low-latency architecture for AES in the context of Cipher-based Message Authentication Code (CMAC). The architecture employs the Block RAM (BRAM) resources available in current FPGAs and utilizes intense parallel processing per cycle. The performance evaluation of the proposed architecture reports a throughput of 3.8 Gbps, a latency of 10 clock cycles for common IoT applications, and resource utilization of 1355 slices and 2 BRAMs.

In [10], Throughput, packet delivery ration, and end-to-end latency are calculated to evaluate the protocol's effectiveness in a mobile network setting where this algorithm has been deployed as part of a hybrid routing protocol. The software used for the simulation is called NS2. We saw an increase in throughput and packet delivery ratio, but it took more time to process.

In [11], they proposed an alternative message. The GPU-optimized Encryption and Authentication Algorithm (MEAA) is suggested. The dynamic key-dependent system is the basis for its one-round encryption and authentication features. The suggested method achieves a throughput of over 580 GB/s in experiments using the GPU Tesla A100.

In [12], In order to facilitate 2-block parallel operation for verification/decryption, they suggest a different implementation technique for serial MAC modes and serial authenticated encryption (AE) modes. The integrity and effectiveness of the original features are preserved in our suggestion. It's new but straightforward, and it may be applied to a broad variety of pre-existing modes, including CMAC and CCM, two recommendations from the NIST. By presenting a number of case studies with actual software implementations, we prove that our concept works.

In [13], They detail a message authentication algorithm (UMAC) that is roughly twice as fast as times previously reported for the universal hash-function family MMH and an order of magnitude faster than current practice (e.g., HMAC-SHA1). UMAC is able to reach these speeds because to the usage of a newly developed family of universal hash functions called NH and a design that makes efficient use of SIMD parallelism. Standard primitives, such as a block cipher or cryptographic hash function, are used to do the "cryptographic" work in UMAC; no new heuristic primitives are built.

In [14], they offer an Authentication-based Matrix-transformation with Parallel-encryption implemented on an asynchronous Multicore Processor (AMP-MP) to achieve a high throughput while remaining safe using the Advanced Encryption Standard with Counter with Chaining Mode (AES-CCM). Our proposed AMP-MP has four important features.

According to the current cryptographic literature, most cryptographic methods need numerous rounds to guarantee the necessary degree of security is achieved. The effectiveness of encryption and decryption methods suffers as a result of this method. However, using parallelism has recently emerged as a practical method to speed up cryptographic software. But other methods fall short of tapping into parallel computing's full potential by failing to take use of its inherent strengths. In light of these findings, the current investigation presents a new message authentication approach tailored to parallel multicore CPUs. To avoid the performance issues that come with more complex encryption methods, this approach just uses a single round of encryption. The suggested method greatly improves the application's performance by taking use of parallelism's computational and communication benefits. It is important to remember that this development helps cryptography since it solves the computational problems that plagued earlier approaches. The authentication process may be optimized for both computation and communication by using a parallel architecture. Therefore, this study makes a significant addition to the field of cryptography by providing a workable strategy for increasing the efficiency of cryptographic applications.

**Table 1.1: Comparison between the Parallel Authentication Methods**

| Description | Year | Parallel platform | Length of key | Type of key | Number of rounds | Security level | Throughput |
|---|---|---|---|---|---|---|---|
| They work presented proposes a concurrent SHA-256 implementation for hashing hundreds of messages on the Sunway TaihuLight SW26010 many-core processor, one of the world's fastest and most powerful computers. | 2023 | Many-core CPU | 256 | Symmetric key | 16 rounds | high | 7.29 GB/s |
| They describe an effective Karatsuba Ofman Algorithm (KOA) based Galois Field (GF) multiplication hardware implementation for | 2012 | Multicore CPU | _ | _ | _ | high | 113.8 GB/s |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| parallel-pipelined GHASH. | | | | | | | |
| They demonstrated a method that was more suitable for GPU capacity and concept. The dynamic key-dependent system, which includes one rozund of substitution and diffusion methods, is the foundation of this system. | 2022 | GPU | (128,192 and 256) bits | Symmetric dynamic key | 1 round | high level of performance and security | more than 400GB/s |
| They have introduced parallel cipher-based message authentication code, a novel authentication technique (PCMAC). The suggested technique has a parallel design and is effective for applications requiring high throughput. | 2019 | Multicore CPU | 128 bits | Symmetric key | _ | high | 2.99 GB/s |
| Presents a CMAC design based on AES with little latency. Each iteration of the design makes use of massively parallel computing and the BRAM available in contemporary FPGA. | 2019 | CPU | (64 and 128 )bits | A symmetric key | 10 round | high | 3.8Gbps |
| Throughput, packet delivery ration, and end-to-end latency are calculated to evaluate the | 2015 | Multi-core processor | 256 bit | Symmetric key | - | Large security | 2.025 GB/s |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| protocol's effectiveness in a mobile network setting where this algorithm has been deployed as part of a hybrid routing protocol | | | | | | | |
| They proposed an alternative message. The GPU-optimized Encryption and Authentication Algorithm (MEAA) is suggested. The dynamic key-dependent system is the basis for its one-round encryption and authentication features. | 2023 | GPU | - | A symmetric key | 1 round | high | over 580 GB/s |
| They presented a novel, straightforward technique of implementation for serial modes. More attention is paid to serial MAC (authentication) and AE modes, in which each input block must be handled in turn in order to maintain compatibility with older protocols. | 2022 | multicore CPU | 128 bits | Symmetric key | 2 round | Large security | 1.52 GB/s |
| They detail a message authentication algorithm (UMAC) that is roughly twice as fast as times previously reported for the universal hash-function family | 2006 | CPU | (32 and 64) bits | Symmetric key | 3 round | High security | 2.9 GB/s |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MMH and an order of magnitude faster than current practice (e.g., HMAC-SHA1). | | | | | | | |
| They present an Authentication-based Matrix-transformation cum Parallel-encryption implemented on an asynchronous Multicore Processor (AMP-MP) to achieve a high throughput and still safe Advanced Encryption Standard based on Counter with Chaining Mode (AES-CCM). Our proposed AMP-MP has four major .features | 2018 | Multi-core | 128 bits | A symmetric key | 10 round | high security | 13.54 GB/s |
| We offer a novel approach for simultaneous message authentication on a multicore CPU. The proposed message authentication technique encrypts the plain message using two PNGRs and two replacement boxes. The one-round operation of the encryption process makes it a quick method to encrypt and decode message blocks. | | Multi-core | 256 bits | Symmetric key | One round | Very high security | 25 GB/s |

## 1.6 Thesis Outline

This thesis is structured as follows:

❖ Chapter Two encompasses a comprehensive examination of the theoretical framework employed in research, encompassing diverse aspects such as encryption, its various types, encryption algorithms, authentication, and the manifold applications of authentication methods, hash functions, parallel computing, MPI, sensitivity tests, and performance measures.

❖ Chapter Three addresses the practical framework employed in the research endeavor through a comprehensive examination of the designed and implemented programs, which aimed to attain optimal outcomes pertaining to speedup, execution time, and throughput.

❖ Chapter Four shows the experimental outcomes of the suggested approach.

❖ Chapter Five shows the conclusion and future work.

# CHAPTER TWO

## THEORETICAL BACKGROUND

## 2.1 Introduction

Encryption, authentication, hash functions, and parallel computing are interconnected concepts that play crucial roles in the field of computer science and information security. Understanding their relationship is essential for ensuring data confidentiality, integrity, and overall system security. Encryption is a technique used to transform data into a form that is unreadable to unauthorized individuals [15]. It involves using an encryption algorithm and a key to convert plain text into ciphertext. Authentication, on the other hand, verifies the identity of a user or entity. It ensures that the communicating parties are who they claim to be. Authentication mechanisms often involve the use of cryptographic techniques such as digital signatures and certificates to establish trust and prevent impersonation or tampering with data. Hash functions are mathematical algorithms that take an input (message or data) and produce a fixed-size output, known as a hash value or digest. These functions are designed to be fast and efficient, and they provide data integrity [1].

Parallel computing refers to the execution of multiple computational tasks simultaneously, leveraging multiple processing units or cores to achieve faster and more efficient processing. It is particularly relevant in the context of encryption, authentication, and hash functions because these operations often involve computationally intensive tasks. Parallel computing techniques can significantly speed up these operations, making them more practical for real-time applications. In practice, encryption algorithms, authentication protocols, and hash functions are commonly used together to provide a comprehensive security framework [16]. For example, encryption can be combined with authentication mechanisms to ensure that only authorized users can access encrypted data. Hash functions can be used to verify the integrity of transmitted data, ensuring that it has not been altered during transmission. Parallel computing techniques can enhance the performance of these operations, enabling faster and more secure processing of large volumes of data[1][15].

## 2.2 Cryptography

The field of study known as cryptography focuses on finding ways to keep information private while still ensuring its safety. The techniques of encryption that are being used to protect websites, messages, and other material in our increasingly digital world may trace their origins back thousands of years.in [17], even if we consider it to be of a basic nature now, the origins of this science, which is still in the process of progressing, can be traced back to the earliest times when communication first began to emerge.

### 2.2.1 Encryption and Keys

Data may be encrypted before being stored or sent to ensure that only authorized individuals are able to view and make use of the information. Data may be encrypted or decrypted using symmetrical or asymmetric encryption methods, which utilize one or more keys according to status and require conversion to encrypted form. This is possible because of the flat state in which the data is now stored. In the case of symmetric encryption, the same key is used for both the encrypting and decrypting of messages or data. In asymmetric encryption, a public key is utilized as the encryption key, while a private key is kept in order to solve the problem of decryption [17][18].

### 2.2.2 Encryption Types

There are two primary encryption techniques utilized in modern-day security protocols. There are two types of encryption: symmetric encryption, in which the same key is used for both encryption and decryption, and asymmetric encryption, in which separate secret (private) keys are used for public decoding [18].

### 2.2.2.1 Symmetric Encryption

To encrypt data or regain access to its unencrypted form, the same key or passwords are used. Information is encrypted and decrypted using the Secret Key. These keys, which may be anywhere from 128 bits to 256 bits in length, are also known as encryption keys or shared keys since they need knowledge from both the sender and the receiver. In [19], since it is more convenient for humans to remember passwords than binary data, most programs make use of them as keys. AES (AES-128, AES-192, and AES-256) is a broadcast symmetric encryption technique used in contemporary cryptography. See Figure 2.1

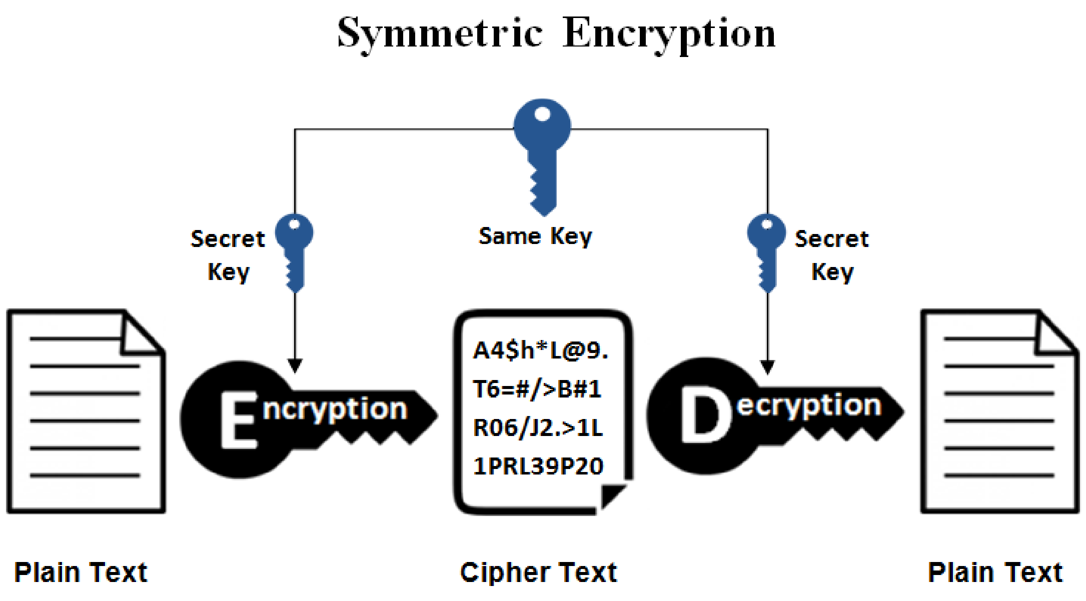


Figure (2.1): Symmetric Encryption [19]

### 2.2.2.2 Asymmetric Encryption

In asymmetric encryption systems, a public key is used to encrypt data, and the resulting encrypted private key is used to decode and authenticate the original data. It is only with the matching private key that the public key's encryption may be decrypted. The result of an encryption procedure is a binary sequence that is unintelligible without the corresponding decryption key. In [20], typically, public key encryptions can only encrypt communications of a small size, whereas a combination of symmetrical and asymmetric encryption is required to encode portable document format (PDF) files or bigger messages. RSA and ECC are the most well-known asymmetric encryption algorithms. See Figure 2.2

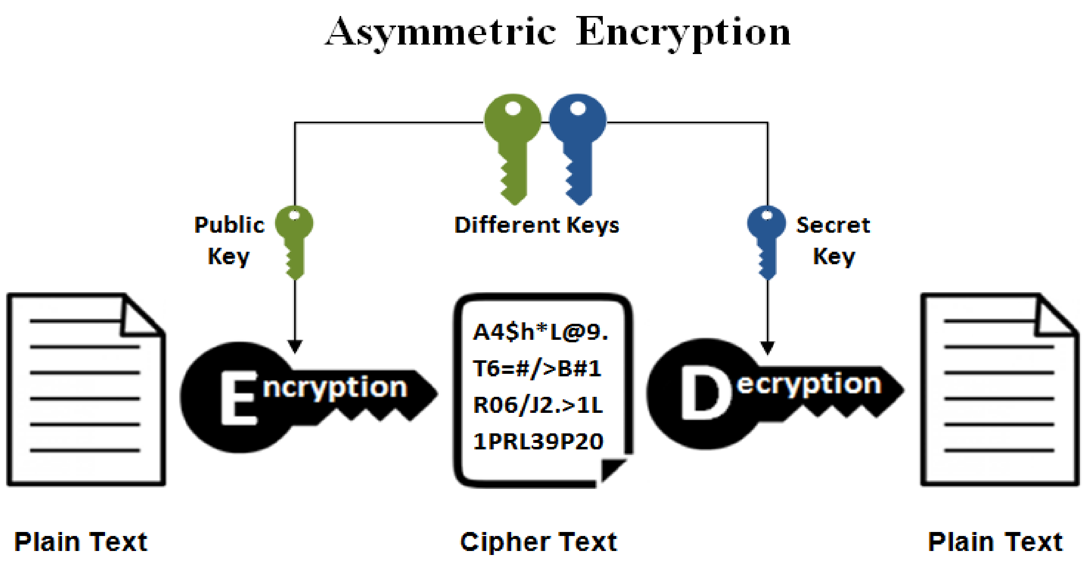


Figure (2.2): Asymmetric Encryption [20]

### 2.2.3 Encryption Algorithms

In the realm of securing the digital transmission of sensitive information, the imperative necessity arises for the implementation of encryption techniques. The process of devising encryption algorithms involves the systematic execution and meticulous documentation of mathematical procedures. In the endeavor of encrypting electronic data or numerical values, a diverse array of block ciphers finds application [21].

### 2.2.3.1 Advanced Encryption Standard (AES)

AES is a symmetric encryption algorithm widely used for securing sensitive information. It uses a block cipher with key sizes of 128, 192, or 256 bits and operates on fixed-size blocks of data [22]. See Figure. 2.3

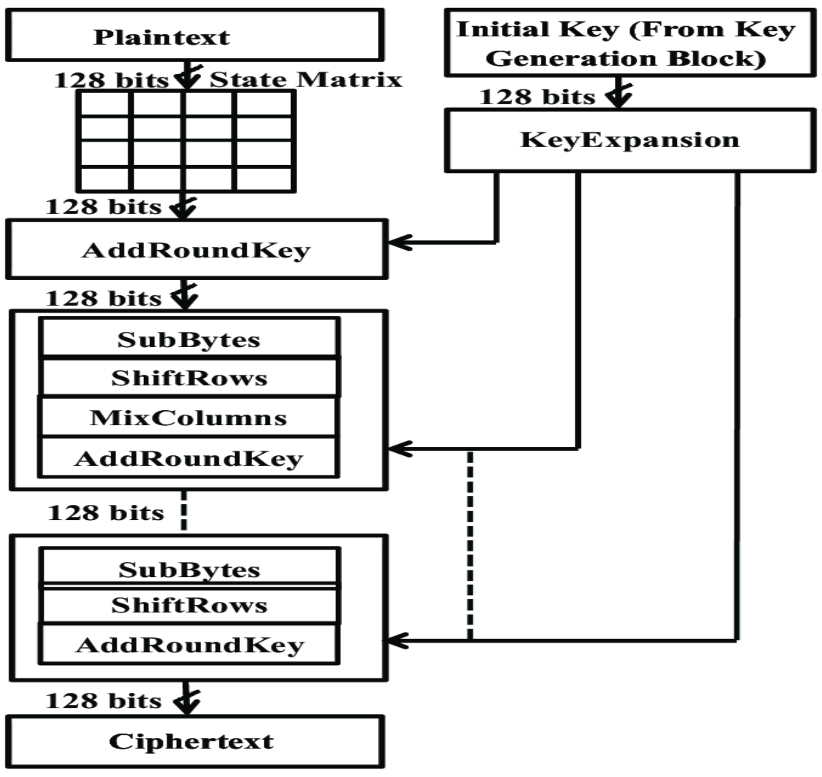


Figure (2.3): AES Encryption [22]

### 2.2.3.2 RSA

RSA is an asymmetric encryption algorithm named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman. It relies on the difficulty of factoring large prime numbers. As shown in Figure 2.4, RSA is primarily used for key exchange and digital signatures [23].

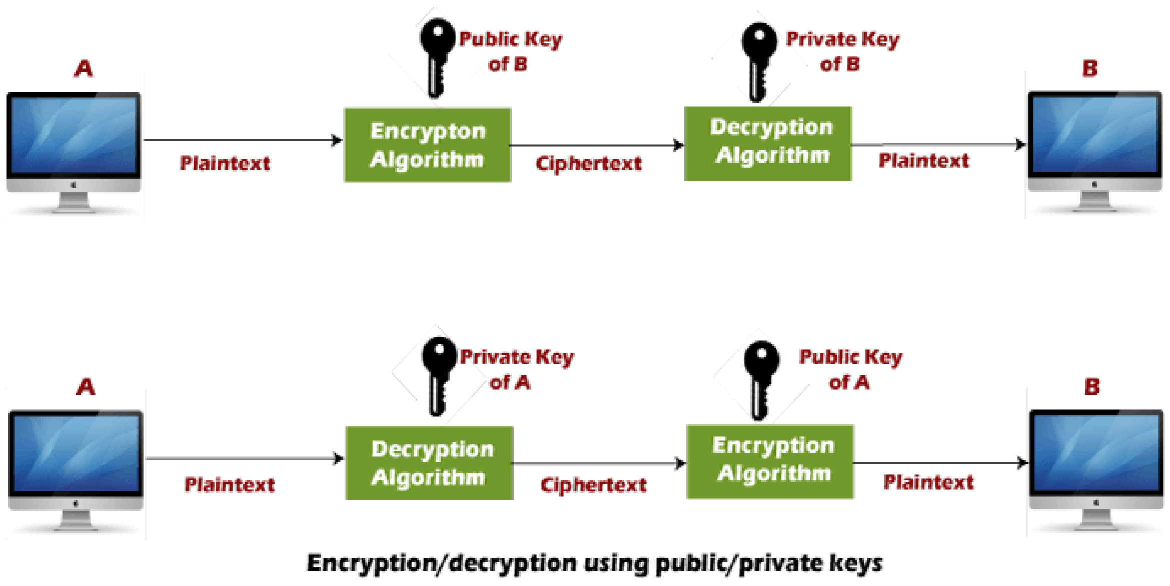


Figure (2.4): RSA Encryption [23]

### 2.2.3.3 Triple Data Encryption Standard (3DES)

It is a symmetric encryption algorithm that applies the DES algorithm three times to each data block. As shown in Figure 2.5, it provides increased security by using multiple encryption operations, but it is slower than AES and is gradually being phased out [24].

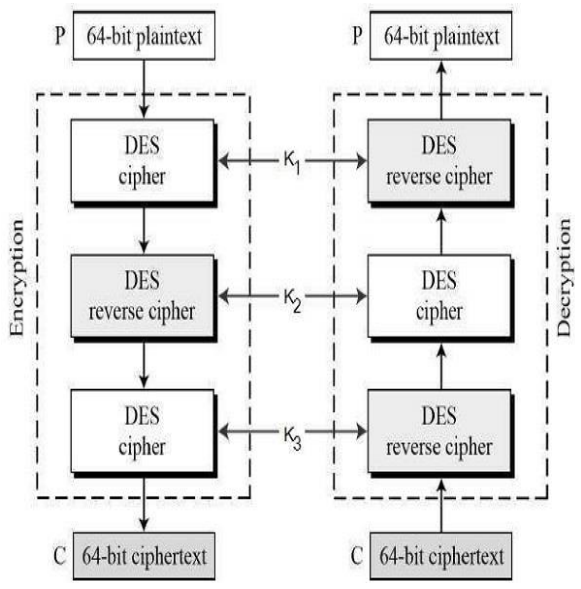


Figure (2.5): 3DES Encryption and Decryption [24]

### 2.2.4 Lightweight Cryptography

Lightweight cryptography refers to a specific area within the field of cryptography that focuses on developing cryptographic algorithms, protocols, and schemes that are highly efficient in terms of resource usage. These lightweight cryptographic solutions are designed to be implemented on resource-constrained devices, such as low-power microcontrollers, RFID tags, wireless sensors, and other embedded systems with limited processing power, memory, and energy [25].

#### 2.2.4.1 Speck Lightweight Cryptography

One of the most efficient block cipher algorithms is Speck. It's flexible in terms of both block and key size [26]. While there are a number of lightweight block ciphers, the vast majority of them are built for a specific platform and are not optimized for use on many devices. Speck's goal is to enable a wide variety of implementations on a particular platform in a wide range of devices running lightweight applications [27], while still meeting the need for safe and high-performance computing on hardware and software platforms. The Speck represents a *wn*-bit key as a 2n-bit block. *2n/wn*, where n is a positive integer between 16 and 64 bits long and m is the minimum length of the key. *m* is version-specific and must be in the range (2, 3), [26], [27]. For example, in Speck 64/128, a *key* 128 is a four-part (32-bit) word expressing in *k=(l[2], l[1], l[0], k[0])* and is used to encrypt 64 block size divided two-part (32-bit) words representing in (*x*, *y*). When Beaulieu and co. See Figure 2.6

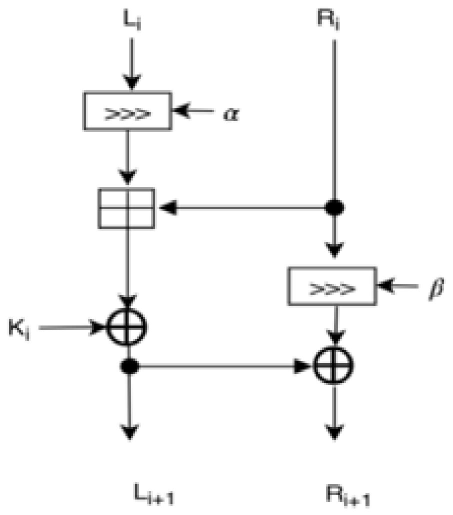


Figure (2.6): Lightweight Speck Cipher [26]

### 2.2.4.2 Simon Cryptography

During the month of June 2013, the National Security Agency (NSA) [28] made the Simon block cipher family available for public use. Simon is a lightweight block cipher family. When compared to its brother method, Speck, which has been tuned for performance in the development of software, Simon has been tailored for performance when it comes to implementing hardware. During 2011, it began working on the Simon cipher as well as the Speck cipher. According to the organization, several departments and agencies within the United States federal government will need a cipher that can successfully operate on a large number of devices connected to the Internet of Things while preserving a satisfactory degree of security. The 10 Speck instances were developed with exceptional hardware and software performance in mind, with a specific attention placed on the microcontroller performance of the instances. Make sure that all of the many sorts of Speck that Speck produces have the exact same name. For example, the term "Speck 96/144" refers to the Speck block cipher that has a block size of 96 bits and a key size of 144 bits [29]. The Simon 2n block cipher is characterized by having a word of length n (and thus, a block of length 2n), where n must be one of the following values:

In Chapter Two, the Theoretical Background, the numbers 18, 16, 24, 32, 48, or 64 are discussed. The term "Simon *2n/mn*" refers to the Simon *2n* algorithm modified to use an m-word (*mn*-bit) key. For example, the form of Simon known as "Simon 64/128" is distinguished by the fact that it employs 128-bit keys and 64-bit plaintext blocks. In each and every instance of Simon, the well-known Feistel rule of motion is implemented. This method was designed to be exceedingly hardware compact and easy to serialize at several layers, all without sacrificing the system's processing performance.

### 2.2.4.3 LEA Cipher

The Low-power Encryption Algorithm, or LEA, was developed in 2013 by the ETRI Attached Institute [30]. Due to its basic Addition-Rotation Exclusive-Or (ARX) and non-S-box architecture, it provides outstanding performance in both software and hardware settings. This is because it was designed without an S-box. LEA is a block cypher of 128 bits and 32 bits for each word. One has a choice between many different security settings, the most common of which are 128-bit, 192-bit, and

256-bit encryption. The necessary number of rounds is 24, and it varies depending on the size of the key: 128 bits, 192 bits, or 256 bits. The technique incorporates both encryption and decryption operations, in addition to key scheduling procedures.

The new block cypher LEA should be considered. It has key sizes of 128 bits, 192 bits, and 256 bits, and its block size should be 128 bits. It allows fast software encryption on CPUs that are designed for general use. This experiment demonstrates that LEA is superior to AES on machines using Intel, AMD, ARM, and Cold Fire. It is also possible to reduce the size of the code using LEA [31].

South Korea came out with a 128-bit block cypher that they termed the Lightweight Encryption Algorithm (LEA) in 2013. This was done in 2013 so that privacy may be protected in high-speed contexts such as large data and cloud computing as well as lightweight contexts such as Internet of Things devices and mobile devices [30]. LEA is capable of supporting keys with lengths of 128, 192, and 256 bits respectively. LEA encrypts data 1.5–2 times quicker than AES in a number of software applications, depending on the context.

The LEA algorithm, which has been validated by the Korean Cryptographic Module Validation Programmer (KCMVP) (KS X 3246), has been selected as the national standard for cryptography in the Republic of Korea [31].

### 2.2.4.4 RC4 Cryptography

RC4 (Rivest Cipher 4) is a symmetric stream cipher widely used in cryptography for encrypting data. It was designed by Ron Rivest in 1987 and is known for its simplicity and speed, making it suitable for various applications, especially in older systems.RC4 operates by generating a pseudorandom stream of bits (keystream) based on a variable-length key. This keystream is then combined with the plaintext using bitwise XOR (exclusive OR) to produce the ciphertext. To decrypt the ciphertext, the same keystream is generated using the same key and XORed with the ciphertext, resulting in the original plaintext. The key length used in RC4 can vary, typically ranging from 40 to 2048 bits, although it's most common use was with a 128-bit key. RC4 does not require padding, as it generates a keystream as long as the plaintext, and the XOR operation ensures that both encryption and decryption are symmetric processes [32].

## 2.3 Authentication

In order to get entry to a protected system, a user or piece of software must go through a procedure known as authentication. After authenticating, the user has access to all the resources, therefore protecting their security and privacy is paramount. When people and their belongings are safe from harm, it ensures their safety and respect as members of society. The term "privacy" is used to describe a scenario in which private information, transaction data, or communication pertaining to a person is protected from disclosure to other parties. As our lives grow increasingly intertwined with digital devices, protecting our personal information becomes more challenging [33].

### 2.3.1 Type of Authentication

In this section, the most important authentication methods will be discussed.

### 2.3.1.1 Password Authentication

The petitioner must use his memory for this kind of authentication. This strategy is split into two stages. As shown in Figure 2.7, the requester must first input the username and then the password. The password is the supplicant's own, private string of letters and digits [34].

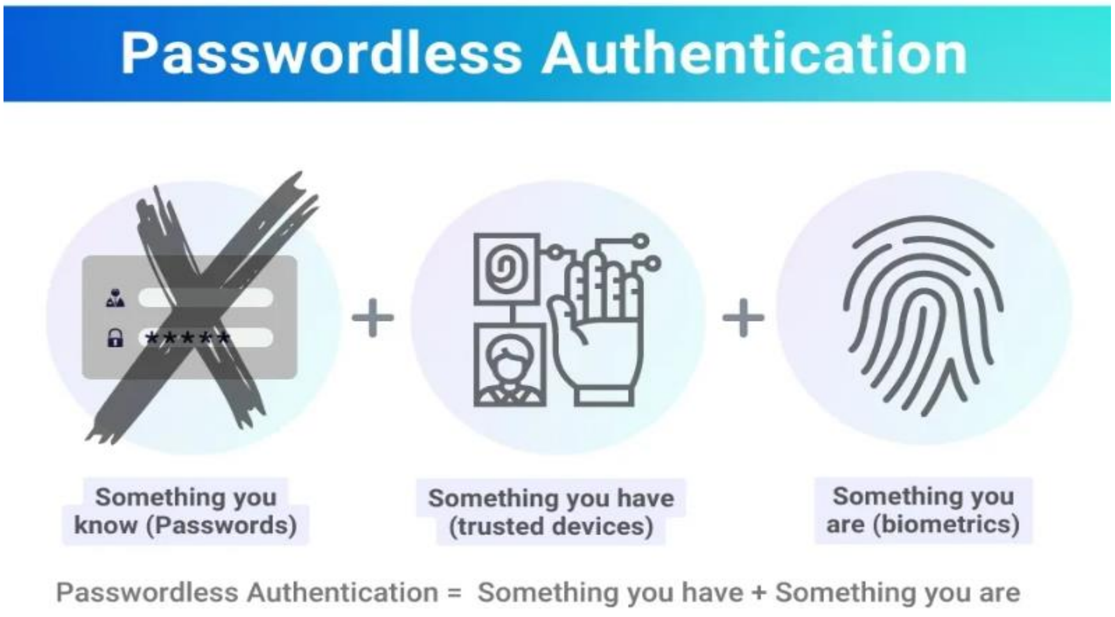

Passwordless Authentication
*****
+
+
Something you know (Passwords)
Something you have (trusted devices)
Something you are (biometrics)
Passwordless Authentication = Something you have + Something you are

Figure (2.7): Password Authentication [34]

### 2.3.1.2 Biometric Authentication

In the context of biometric authentication, biometric data is employed. Electroencephalography (EEG), body movements, and Electrooculography (EOG) measurements are just a few examples of the various data types that may be employed in this kind of verification. In [35]. Due to its singularity, EEG data is trustworthy. See Figure 2.8

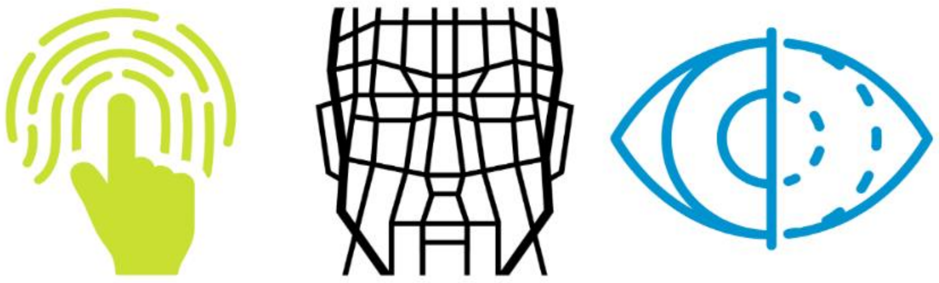


Figure (2.8): Biometric Authentication [35]

### 2.3.1.3 Multi-model Authentication

Using this approach, users may perform authentication via a mix of two or more methods. Because an attacker now has to circumvent many layers of defense, the system's security has been bolstered. As shown in Figure 2.9, one model that fits this description is the RubikBiom that has been created in this area [36]. The user's authentication biometrics are under the control of the password put into the Rubik's cube.

Figure (2.9): Multi-model Authentication [36]

### 2.3.1.4 Public Key Infrastructure (PKI)

PKI is a system that uses public key cryptography to authenticate the identity of users and devices. It involves the use of digital certificates, which are issued by trusted certificate authorities and used to verify the authenticity of public keys [37].

### 2.3.1.5 Token-based authentication

Tokens are small hardware devices or software applications that generate one-time passwords or passcodes. In [38], these passcodes are time-based or event-based and are used in addition to a regular username and password for authentication.

## 2.3.2 Uses of Authentication

1. User authentication: Authentication is commonly used to verify the identity of users accessing online services, such as email accounts, social media platforms, online banking, and e-commerce websites.
2. System authentication: Systems and devices authenticate each other to establish secure communication channels. This is crucial for protecting sensitive information and ensuring data integrity.
3. Network authentication: Authentication is used in network environments to control access to resources, such as Wi-Fi networks, VPNs (Virtual Private Networks), and remote access systems.
4. Application authentication: Applications often require authentication to ensure that only authorized users or entities can access their functionalities and data.
5. API authentication: APIs (Application Programming Interfaces) often implement authentication mechanisms to control access to their services and protect sensitive data exchanged between applications.

Overall, authentication is a critical security mechanism used in various contexts to verify identities, control access, and protect sensitive information and resources [39].

## 2.4 Hashing Method

Hashing is a process of converting data (such as a string or file) into a fixed-size numerical value, called a hash or hash code. This numerical representation is typically generated using a hash function, which takes the input data and produces a unique hash value. In [40], hashing is widely used in computer science and cryptography for various purposes, including data storage, data retrieval, and data integrity verification.

### 2.4.1 Types of Hashing

The main difference between cryptographic and non-cryptographic hashing lies in their security goals and design principles. Cryptographic hashing is designed to provide strong security guarantees, while non-cryptographic hashing prioritizes efficiency and speed for non-security-related applications [41].

#### 2.4.1.1 Cryptographic Hashing

Cryptographic hash functions are designed to be secure and resistant to attacks. They produce a fixed-length hash that is unique to the input data, meaning even a small change in the input results in a significantly different hash value. Examples of cryptographic hash functions include MD5, SHA-1, SHA-256, and SHA- Cryptographic hashing is commonly used in password storage, digital signatures, message integrity checks, and data authentication [41].See Figure 2.10

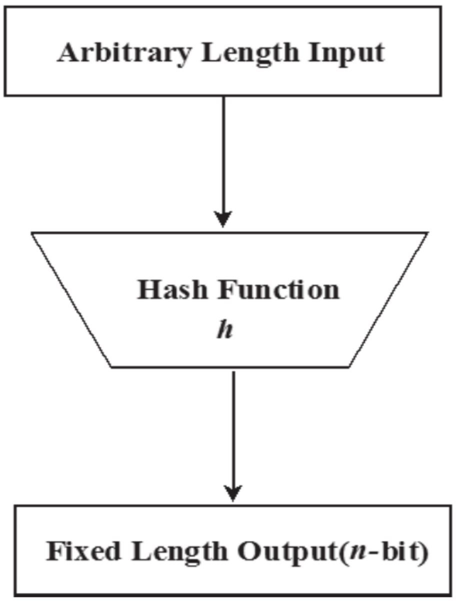


Figure (2.10): Cryptographic Hashing [41]

#### 2.4.1.2 Non-Cryptographic Hashing

Non-cryptographic hash functions are primarily used for data retrieval and indexing purposes. While they are not designed for security, they provide efficient and fast hash computations. In [42], non-cryptographic hashing algorithms aim to evenly distribute the data across a fixed range of hash values to minimize collisions (different inputs producing the same hash). Examples of non-cryptographic hash functions include Murmur Hash, Jenkins hash, and FNV hash. Non-cryptographic hashing is used in hash tables, data caching, indexing structures, and data duplication.

### 2.4.2 Types of Hash Function Algorithms

A hashing algorithm is a systematic and deterministic procedure used in computer science and cryptography to convert input data of arbitrary length into a fixed-size output, known as the hash value or hash code. The primary purpose of a hashing algorithm is to efficiently represent and manage data, allowing for rapid data retrieval, comparison, and verification [43]. We will mention some types of hashing algorithms.

### 2.4.2.1 MD5 (Message Digest Algorithm 5)

Ronald Rivets came up with the Message-Digest method 5 (also known as MD5) in 1991 with the intention of replacing the previous hash function, which was MD4, which cryptanalysts were able to effectively attack. The MD5 method takes data in the form of a message that may be of any size as input and produces a message digest that is 128 bits in length. One of the programmers that is used to ensure that the messages that have been transmitted do not undergo any alterations while they are on the network is called MD5. The MD5 algorithm is in the process of taking a message that has a variable length and changing it to a "fingerprint" or "essence of the message" that has a fixed length of 128 bits. This process is described in more detail below. This "fingerprint" cannot be reversed to reveal the message; to put it another way, no one can see the contents of the "fingerprint" generated by the MD5 algorithm. MD5 has been used in a wide variety of security programmers, and it is also often used as a tool for determining whether or not a file has been tampered with. Message Digest 5, often known as MD5, is one of the one-way hash algorithms that is used the most frequently [43].

### 2.4.2.2 SHA-1 (Secure Hash Algorithm 1)

SHA-1, short for "Secure Hash Algorithm 1," is a cryptographic hash function that was designed to produce a fixed-size hash value from an input of variable length. It was developed by the United States National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) in 1993 as part of the Digital Signature Algorithm (DSA) standard.

SHA-1 generates a 160-bit (20-byte) hash value, commonly represented as a 40-digit hexadecimal number. The hash value is intended to uniquely represent the input data, ensuring that even a minor change in the input will result in a significantly different hash value. [44].

#### 2.4.2.3 SHA-256 (Secure Hash Algorithm 256)

SHA-256, which stands for Secure Hash Algorithm 256, is a cryptographic hash function that is part of the SHA-2 (Secure Hash Algorithm 2) family of hash functions. It was developed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) in 2001.SHA-256 takes an input message of variable length and produces a fixed-size hash value of 256 bits, or 32 bytes. The resulting hash value is typically represented as a 64-digit hexadecimal number. Like other cryptographic hash functions, SHA-256 is designed to be a one-way function, meaning it is computationally infeasible to reverse the process and obtain the original input from the hash value. [5].

#### 2.4.2.4 SHA-3 (Secure Hash Algorithm 3)

SHA-3, which stands for Secure Hash Algorithm 3, is the latest member of the Secure Hash Algorithm family of cryptographic hash functions. It was designed as the successor to SHA-1 and SHA-2 and was developed by a competition organized by the National Institute of Standards and Technology (NIST). The competition, known as the SHA-3 competition, was launched in 2007 to select a new standard cryptographic hash function that would offer improved security and performance. [45].

## 2.5 Parallel Computing

Parallel computing refers to the simultaneous execution of multiple computational tasks or operations, with the goal of reducing the overall execution time and improving performance. It involves breaking down a complex problem or task into smaller sub problems that can be solved concurrently. Traditionally, computers have used sequential processing, where instructions and operations are executed one after another. In contrast, parallel computing utilizes multiple processing units or cores to perform computations simultaneously [46]. See Figure 2.11

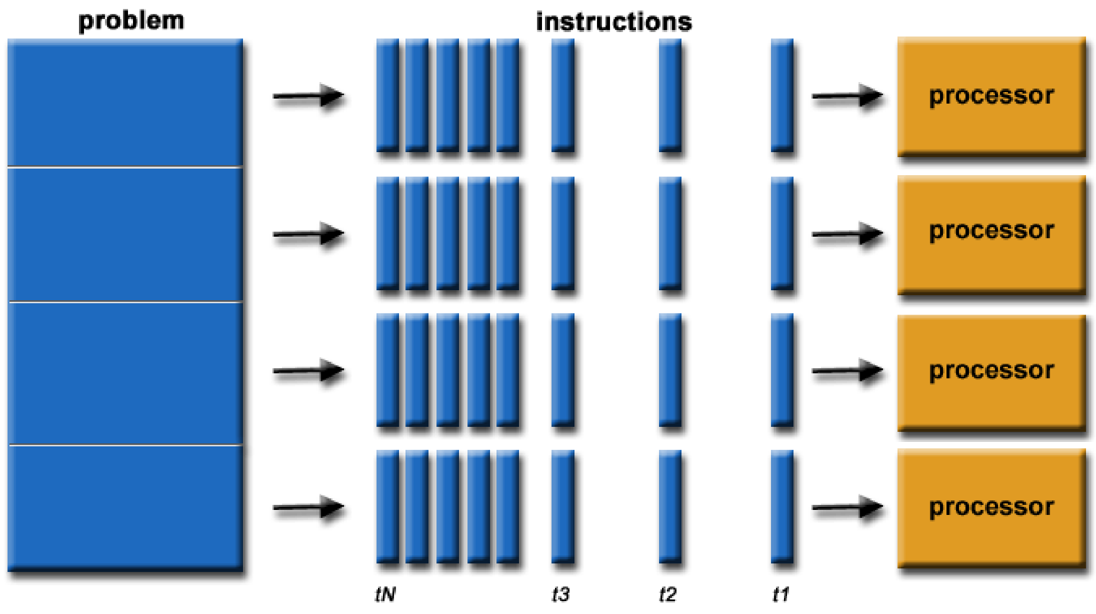


Figure (2.11): Parallel Computing [46]

### 2.5.1 Levels of Parallelism

Parallelism refers to the concept of breaking down a task into smaller subtasks that can be executed simultaneously, thereby increasing overall efficiency and reducing the time required to complete the task. Levels of parallelism, also known as levels of concurrency or levels of parallel execution, refer to the different layers or stages at which parallelism can be applied in a system or computation [46].

#### 2.5.1.1 Instruction-level Parallelism (ILP)

This level of parallelism involves executing multiple instructions simultaneously within a single processor core. It leverages techniques like pipelining, superscalar execution, and out-of-order execution to overlap the execution of multiple instructions, taking advantage of independent operations [47].

#### 2.5.1.2 Thread-level Parallelism (TLP)

This level of parallelism involves executing multiple threads or processes simultaneously across multiple processor cores or multiple computing nodes in a distributed system. Each thread or process works on a different part of a task or handles different independent tasks altogether. TLP is typically used in multi-core processors, clusters, or multi-node systems [48].

#### 2.5.1.3 Task-level Parallelism (TALP)

This level of parallelism involves breaking down a larger task into smaller, independent tasks that can be executed in parallel. These tasks can be distributed across multiple computing nodes or processors and can be executed concurrently. TALP is commonly utilized in high-performance computing (HPC) systems, grid computing, or distributed computing environments [49].

### 2.5.2 Shared and Distributed Parallel Memory Architectures

Shared and distributed parallel memory architectures are two different approaches to organizing and accessing memory in parallel computing systems [50].

#### 2.5.2.1 Shared Memory Architecture

In a shared memory architecture, multiple processors or computing units share a single, global memory address space. This means that all processors can access any location in the memory directly, as shown in Figure 2.12. This shared memory can be implemented physically as a single memory module or logically through a memory management system [50].

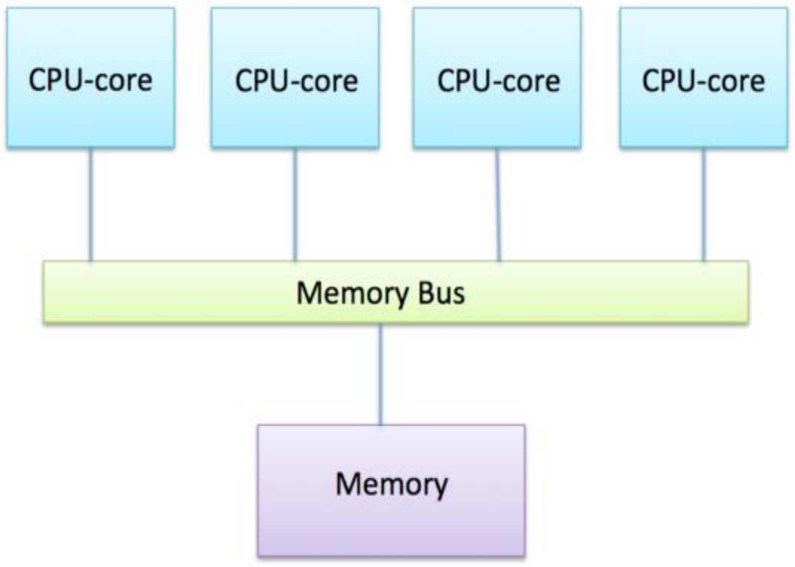


Figure (2.12): Shared Memory Architecture [50]

#### 2.5.2.1.1 Multi-core Processor

A multi-core processor is a type of computer processor that integrates two or more independent processing units, known as cores, onto a single physical chip. Each core within the processor functions as a separate and autonomous processing unit capable of executing instructions independently [51].

The concept behind multi-core processors is to enhance the processing power of a computer system by dividing the workload among multiple cores. This enables parallel processing, where different cores can simultaneously execute instructions, leading to increased performance and improved overall efficiency. Each core within a multi-core processor typically includes its own arithmetic logic unit (ALU), control unit, cache memory, and other components necessary for independent instruction execution, as shown in Figure 2.13. These cores may share certain resources such as memory controllers, cache hierarchy, and system interfaces to optimize performance and reduce duplication of components [51] [52].

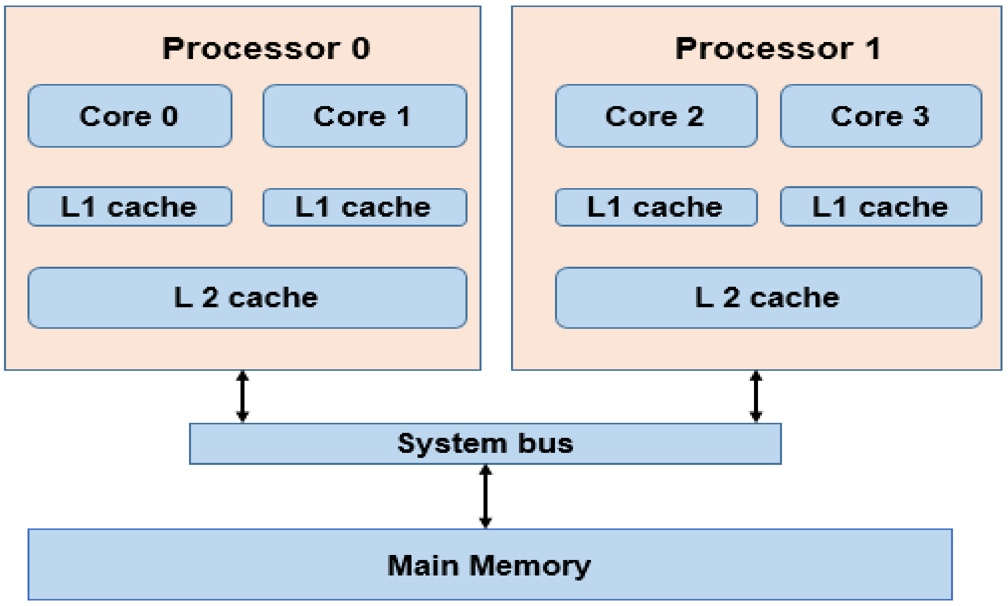


Figure (2.13): Multi-core processor [52]

### 2.5.2.2 Distributed Memory Architecture

Distributed memory architecture, also known as a distributed system, is a type of computer architecture where multiple independent computing nodes or processors are connected via a network. Each node has its own private memory and operates independently, with communication occurring through message passing between nodes, as shown in Figure 2.14. Distributed memory architecture encompasses diverse typologies or paradigms, namely cluster computing, grid computing, cloud computing, and distributed computing [53].

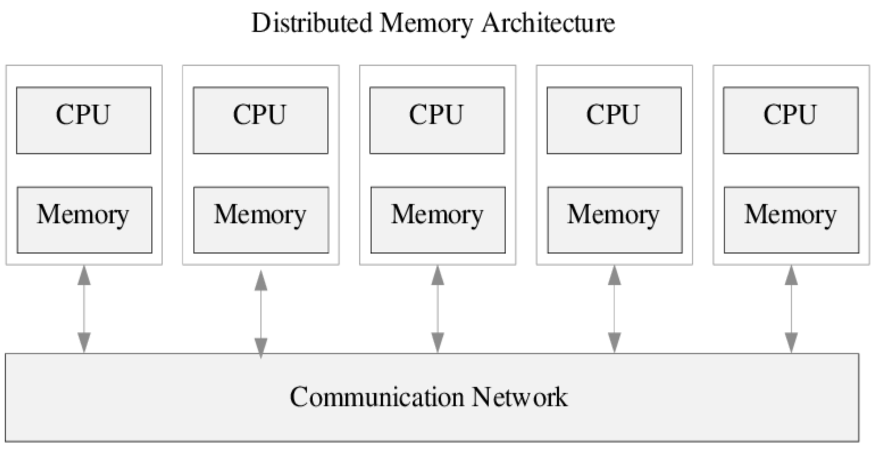


Figure (2.14): Distributed Memory Architecture [53]

## 2.6 Message-Passing Interface (MPI)

The MPI library is an industry- and research-driven standard for communicating across a wide variety of parallel computer systems. It is a standard that establishes the syntax and semantics of a core library procedure that may be used by a wide range of programmers to create portable message-moving applications that run on shared and distributed memory cluster platforms using languages like C, C++, and FORTRAN. The MPI library's send and receive functions allow users to largely manage data flow between groups of communicating nodes. This provides the context for understanding the term "communicator," which is used to characterize a group of processes that may exchange information with one another. Therefore, Chapter Two focuses on the sending and receiving actions between processes. The foundation of communication, parallel hardware and energy efficiency [56]. There is an explicit ranking system between each procedure in this category. Messages may be sent from one process to another by including the sending process's rank and a unique tag. The receiver may then choose to submit a message with a certain tag (or disregard tags altogether) and organize the data as needed.

Point-to-point communications describe interactions between a single sender and a single recipient. In other scenarios, processes may need to talk to one other, such as when a process manager has to relay data to his subordinate processes [57]. The MPI allowed for collective communication, including barrier, broadcast, gathering, and reduction operations, in situations when all processes needed to talk to one other.

In addition, the MPI allows for both synchronous and asynchronous exchanges of data. Unlike synchronous operations, which halt a process until the send/receive operations finish once the message has been delivered to the receiver, asynchronous communications allow the process to continue executing computations even as communication with other processes is still in progress [56].

## 2.7 Security Analysis

The suggested authentication technique is tested against common attacks including statistical, linear, differential, and brute-force attacks to make sure it can withstand them. In this part, we conduct extensive tests to prove that our suggested authentication method is safe and effective. While the proposed approach is applicable to any kind of data

### 2.7.1 Statistical Analysis Tests

Authentication and encryption against statistical attacks need randomness and homogeneity. Analysis of the PDF, histogram, entropy, and correlation between the original and encrypted messages are only some of the statistical integrity checks that are performed.

#### 2.7.1.1 Histogram Analysis

The histogram of the encrypted message has to be normally distributed so that the uniformity of the proposed one-round encryption utilized in the authentication method may be tested. In [58], histogram analysis helps visualize the shape and characteristics of a dataset, providing valuable information about its distribution. It is commonly used in various fields, including data analysis, finance, quality control, and research. By examining the histogram, you can identify trends, identify data clusters or gaps, detect anomalies, and make informed decisions based on the data's distribution.

The general equation for a histogram can be expressed as follows:

$$H\,(i) = N\,(i) \,/\, (A * \Delta x) \qquad (2.1)$$

Where:

- $H\,(i)$ represents the value of the histogram bin at index $i$.
- $N\,(i)$ represents the frequency or count of occurrences of data points falling within the *i-th* bin.
- $A$ is the total number of data points or observations in the dataset.
- $\Delta x$ represents the width or size of each bin in the histogram.

In simpler terms, the equation calculates the relative frequency of data points falling within each bin of the histogram. The count of data points within a particular bin ($N\,(i)$) is divided by the total number of data points ($A$) multiplied by the width of each bin ($\Delta x$). This normalization ensures that the sum of all bin values in the histogram equals 1, representing the probability or relative frequency distribution of the data [58].

#### 2.7.1.2 Uniform Security Test

If the encrypted message is to pass this test, its probability density function (PDF) must be uniform, since it is considered to be the most important of the tests. In [59], each symbol in the ciphertext has a chance of appearing in the ciphertext that is very close to $1/n$, where $n$ is the total number of symbols.

### 2.7.1.3 Information entropy analysis

Entropy analysis, in the context of information theory, is a technique used to quantify the amount of information or uncertainty contained in a dataset or a random variable. It provides a measure of the average information content required to describe or predict the outcomes of the variable.

$$h(m) = -\sum_{i=1}^{h2} p(mi) \log_2 \frac{1}{p(mi)} \tag{2.2}$$

In this equation:

*H(m)* represents the entropy of the random variable m.h2 is the total number of possible outcomes of the random variable *m*. *p(mi)* denotes the probability of the it outcome, *mi*, occurring.

For each outcome mi, we calculate the information content, represented by $\log_2$ (1/*p* (*mi*)). The information content is determined by taking the logarithm (base 2) of the reciprocal of the probability *p (mi).* It measures the amount of information required to represent or predict the occurrence of *mi*. The use of the logarithm allows us to express the information content in bits. We multiply the information content by the probability *p(mi)* of the corresponding outcome. This step ensures that the contribution of each outcome to the overall entropy is weighted by its probability. Outcomes with higher probabilities will have a larger impact on the entropy calculation. We sum up the products obtained from step 2 for all possible outcomes, ranging from *i* = 1 to *h2*. The summation captures the combined contribution of each outcome to the overall entropy [60].

### 2.7.1.4 Correlation

For the proposed encryption system to be foolproof, it is essential that all ties be severed between the sequences of its constituent parts [61]. The cipher scheme is very random when the correlation coefficient is modest (near to zero). The correlation test is performed by picking pairs of adjacent pixels from the original message and their corresponding encrypted ones. The relationship is true for directions in the horizontal, vertical, and diagonal planes. You may calculate the *Rxy* correlation coefficient with the help of the following equations:

$$R_{xy} = \frac{\text{cov(x,y)}}{\sqrt{D(x)\times D(Y)}}$$

Where:

$$\text{cov(x,y)}= \frac{1}{n}\sum_{i=1}^{n}(x_i - E(x))(y_i - E(y)) \qquad (2.3)$$

$$E_{x=}\frac{1}{n} \times \sum_{i=1}^{n} x_i$$

$$D_{x=}\ \frac{1}{n}\times\ (\sum_{i=1}^{n}(x_{i-}\ \text{E(x)})^2$$

In the previous calculations, the correlation coefficient *Rxy* was used to determine the relationship between the x and y sequences. The values of x and y are represented by the symbols *mn* and *yi*, respectively. We performed a correlation analysis of neighboring pixels in both the cypher and the original to demonstrate that the cypher image is fully unique from the original [6].

### 2.7.2 Sensitivity Tests

Differential attacks concentrate on figuring out how two encrypted communications are related when one of the original messages has a tiny modification, such a one-bit difference. In [3], tests for sensitivity should show that the ciphertext changes if the plaintext or key are modified even slightly. The suggested cipher scheme's sensitivity increases as the difference increases.

### 2.7.2.1 Key Avalanche Effect

The critical avalanche test holds considerable significance as it serves as a fundamental assessment. Each data encryption technique exhibits distinct advantages and disadvantages. Hence, it is imperative to be cognizant of both strengths and weaknesses to select the most effective method in addressing specific challenges. Consequently, an investigation of this test becomes an indispensable necessity. An ideal encryption approach should possess the desirable property that even a minor alteration to either the plaintext or the key would lead to a discernible transformation in the ciphertext. In [62], this particular attribute stands as one of the most sought-after characteristics of encryption. When a single bit in either the plaintext or the key undergoes modification, a consequential modification in a significant number of bits in the ciphertext should result. This phenomenon is commonly referred to as the "avalanche effect."

The general equation for avalanche Effect can be expressed as follows:

$$\text{Avalanche effect} = \frac{Number\ of\ flipped\ bitsin\ ciphered\ text}{Number\ in\ bits\ of\ ciphered\ text} \qquad (2.4)$$

### 2.7.2.2 Message Avalanche Effect

Utilizing a distinct dynamic key for each message input, the encryption technique ensures that the ciphertext derived from identical plaintext inputs will invariably differ [62]. The message avalanche effect, commonly referred to as the bit avalanche effect or simply avalanche effect, constitutes a favorable attribute within cryptographic systems, particularly in the context of encryption algorithms. It characterizes the phenomenon where a minute modification to either the input message or the encryption key induces a substantial and unpredictable transformation in the resulting ciphertext. In essence, even a marginal perturbation to the plaintext or encryption key should precipitate a profound and widespread alteration in the ciphertext.

### 2.7.3 Randomness Test

There are several tools and software packages available for conducting randomness tests. Here are some of the commonly used ones:

- PractRand, short for "Practically Random," is a widely used pseudorandom number sequence test suite. It is designed to rigorously evaluate and analyses the quality of pseudorandom number generators (PRNGs) or random number generators (RNGs). The primary purpose of PractRand is to detect potential patterns, biases, or other weaknesses in PRNGs that could lead to predictability or suboptimal randomness in the generated sequences. The PractRand test suite was created by Chris Lomont, and it provides a battery of statistical tests that assess the randomness of a given sequence of random numbers. It uses statistical analysis, focusing on the empirical distribution and correlations of subsequences, to reveal patterns or anomalies that indicate insufficient randomness. The tests in PractRand help identify flaws in PRNGs, which could be crucial in cryptographic applications, simulations, and other scenarios where high-quality random numbers are essential [63].

-TestU01: TestU01 is a widely used library of statistical tests for pseudorandom number generators (PRNGs). It provides a comprehensive set of tests, including various suites of tests such as SmallCrush, Crush, and BigCrush, which differ in terms of test intensity and time requirements [64].

-Diehard and Dieharder: Diehard and Dieharder are sets of statistical tests developed by George Marsaglia. Diehard is a collection of statistical tests for testing the randomness of random number generators, while Dieharder is an extended and enhanced version of Diehard. These tests cover a broad range of randomness properties [65].

-ENT: ENT (Entropy) is a command-line tool that performs various statistical tests on data files. It calculates various measures of entropy, such as arithmetic and compression entropy, and conducts tests like the chi-square test, serial correlation test, and rank order test [66].

-NIST Statistical Test Suite (STS): The NIST STS is a suite of tests developed by the National Institute of Standards and Technology (NIST). It provides a comprehensive set of statistical tests to assess the randomness of binary sequences. The suite includes tests for frequency, block frequency, runs, autocorrelation, and many others [67].

These tools and software packages can be used to evaluate the quality and randomness of random number generators, analyze data sequences, or assess the statistical properties of cryptographic algorithms.

It's important to choose the appropriate tool based on the specific requirements and characteristics of the data being analyzed [67].

## 2.8 Performance Evaluation

In the realm of performance evaluation and comparison, key metrics such as speedup, execution time, and throughput are commonly used to assess the efficiency and effectiveness of systems, algorithms, or approaches. These metrics provide valuable insights into the performance characteristics and relative merits of different options.

### 2.8.1 Execution Time

It is possible for the execution time of parallel cryptographic algorithms to vary greatly depending on a number of different circumstances. These considerations include the exact method that is being used, the quantity of the input data, the hardware platform on which it is being processed, and the unique implementation of the algorithm. During the sequential execution of an algorithm that encrypts or decrypts a message of size *n*, where the time required to encrypt each block is *Ti*, the total amount of time required to encrypt all blocks is calculated as follows:

$$T_{Seq} = \sum_{i=0}^{n-1} Ti \tag{2.5}$$

When tasks are completed in parallel, the total amount of time required to encrypt or decode a message block is determined by the job that takes the longest to complete. The calculation for this is as follows:

$$T_{Par} = Max_{i=0}^{n-1} \; (T_i) \tag{2.6}$$

However, contemporary hardware platforms and software implementations that have been optimized may assist minimize the amount of time required for cryptographic algorithm execution. The execution of certain cryptographic processes, for instance, may be made a lot quicker with the help of hardware-accelerated cryptographic modules or specialized cryptographic co-processors [68].

## 2.8.2 Speedup

Speedup refers to the measure of performance improvement achieved by a system or algorithm when compared to a reference or baseline. It quantifies how much faster a system or algorithm performs a specific task compared to a standard or previous versio-n. In [69], Speedup is typically calculated as the ratio of the execution time of the baseline to the execution time of the improved system or algorithm. A speedup value greater than 1 indicates that the improved system is faster than the baseline, while a value less than 1 suggests the opposite.

$$Speedup=\frac{Tseq}{Tpar} \tag{2.7}$$

Time spent in the execution delay ($Tseq$) for the sequential version, and time spent in the parallel version ($Tpar$) for the same task. However, the value of the acceleration may be represented in different ways, such as a multiplier. 2x, meaning twice the rate at which things are going now. This means the rate will increase by a factor of two. This will provide insight into the method's scalability by evaluating how much the speedup could change with a change in the number of processing units [69].

## 2.8.3 Throughput

Throughput represents the rate at which a system or algorithm can process or handle a certain amount of work over a given period. It measures the system's ability to efficiently process input and produce output. In [36], throughput is typically measured in terms of tasks completed, data processed, or transactions handled per unit of time.

Higher throughput values indicate a system's capability to handle larger workloads or process data more quickly.

$$Throughput\ (GigabitPerSecond) = \frac{Message\ size(Gigabit)}{Execution\ Time(Sec)} \tag{2.8}$$

The aforementioned equation is used by several researchers in the literature; for example, see [70].

## 2.9 Summary

This chapter presents a comprehensive overview of encryption methods, encompassing their application in parallel computing scenarios. It further provides an extensive elucidation on authentication, elucidating various authentication methods and their pertinent applications. The discussion delves into the fundamental concept of hash functions, elucidating their purpose and practical utility. Furthermore, the chapter explores parallel computing in depth, elucidating distinct levels of parallelism and introducing the essential concept of the Message Passing Interface (MPI). Concluding this discourse, the chapter delves into the domain of sensitivity tests and performance measures, shedding light on their significance and practical implementation.

# CHAPTER THREE

# THE PROPOSED SYSTEM

## 3.1 Introduction

In this chapter, the proposed parallel message authentication method is comprehensively expounded, wherein the message size is specified as 256 bits. Subsequently, the message is partitioned into four blocks, with each block having a size of 64 bits. Encryption and authentication procedures are conducted on each individual block, thereby facilitating a comprehensive evaluation of the method's impact in terms of execution time, throughput, and speedup. The exposition further encompasses a detailed elucidation of all employed algorithms, namely splitmix64, xorshift64, Rc4, and dynamic key. A thorough analysis is provided for the functionality and role of each algorithm within the context of the proposed method.

## 3.2 Proposed Parallel Message Authentication Method

The message is broken up into a number of blocks using the suggested technique for authenticating it. The size of each block is equal:

$$X=N/NP$$

Where $N$ is the total size of the message and $NP$ is the total number of processes that are running in parallel. The message's blocks are uniformly spread among all of the processes that are running in parallel. The local Message Authentication Code (MAC) is subsequently derived from the results of the subsequent encryption and authentication operations that are carried out on each block. In order to improve the unpredictability, non-linearity, and confusion of the MAC value, we make use of the PRNG of Splitmix64 and diffusion functions that are implemented by substitution (see Equation 3.1). After these processes, a global MAC is created by applying the parallel asynchronous gathering procedure to all of the MAC values. This results in the generation of the global MAC. As shown in

Figure 3.1, the global MAC may be calculated by applying the XOR operation to the aggregated MAC values that have been acquired from each individual process. This yields the global MAC. In addition, the methods of substitution and splitmix64 are used once again for the final MAC value in order to increase the amount of unpredictability and nonlinearity that it has.

$$MAC\ (b1,\ b2,\ b3,\ b4) = splitmix64\ (substitution\ (c1 \oplus c2 \oplus c3 \oplus c4)) \quad (3.1)$$

Where *b1*, *b2*, *b3*, and b4 are the blocks that make up a plain message and *c1*, *c2*, *c3*, and *c4* are the blocks that have been encrypted by using the one-round function that has been suggested. Authentication functions are shown further in Figure 3. 2, which depicts the function that corresponds to Equation 3.1.

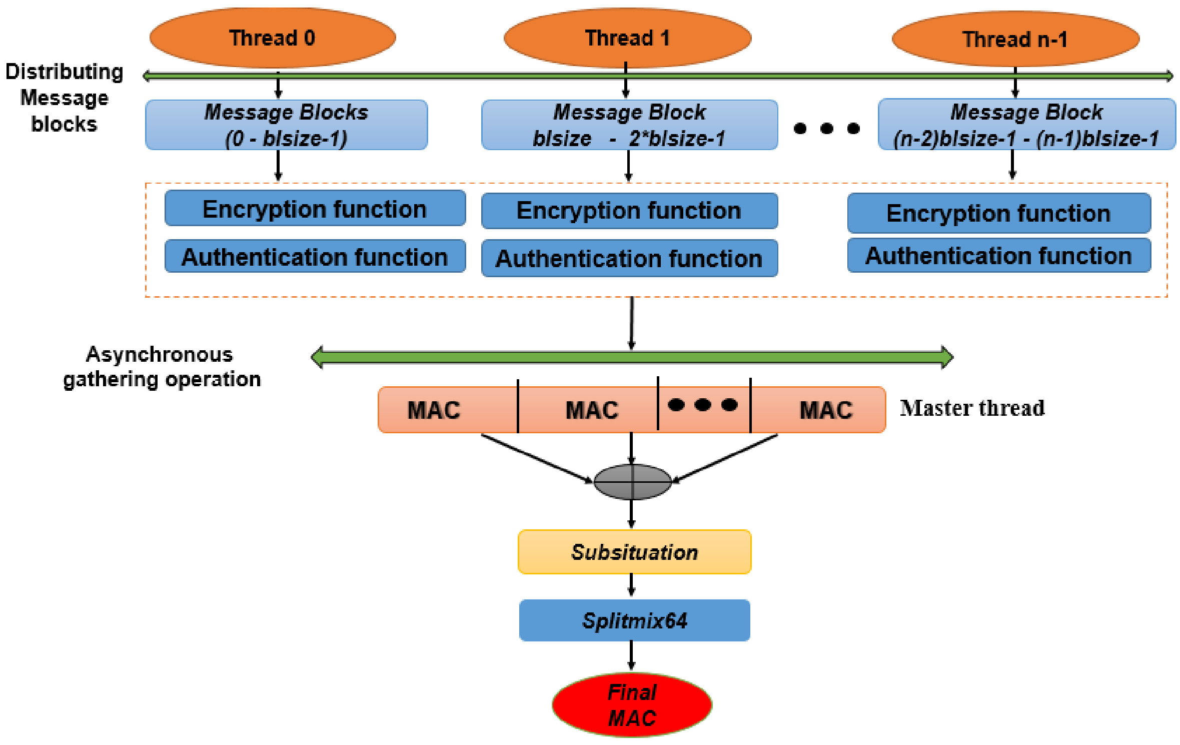

Thread 0
Thread 1
Thread n-1
Distributing Message blocks
Message Blocks (0 - blsize-1)
Message Block blsize - 2*blsize-1
Message Block (n-2)blsize-1 - (n-1)blsize-1
Encryption function
Authentication function
Asynchronous gathering operation
MAC
Master thread
Subsituation
Splitmix64
Final MAC

Figure (3.1): The Structure of the Proposed Parallel Authentication Method

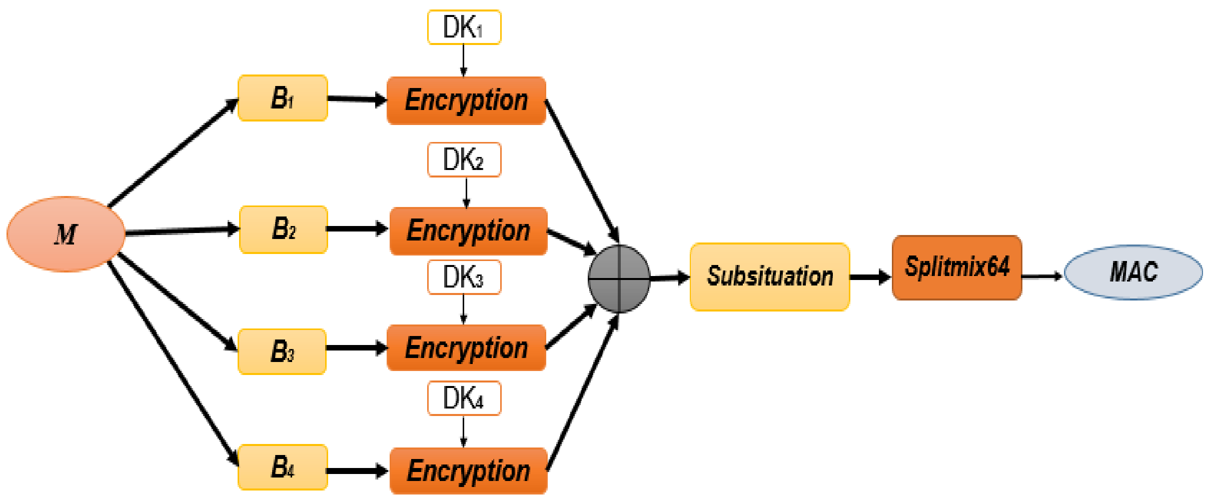


Figure (3.2): The Authentication Function

### 3.2.1 Dynamic Key Generation Method

The dynamic key-dependent approach is the basis for the proposed solution. In this method, a dynamic key (*DK*) is applied in order to construct a collection of dynamic cryptographic primitives. (In addition to a set of *N* seeds, substitution tables are required, with each seed having the potential to be either a 32-bit or 64-bit word.) This dynamic key, referred to as *DK*, is obtained by conducting an XOR operation between an initial vector and *CTR* (counter mode). This key, which must be updated and unique for each transmission, should be obtained this way. This operation is carried out in the way shown in Figure 3.1, and it is stated in the following manner in Equation 3.2:

$$DK = (IV \oplus CTR \oplus key) \qquad (3.2)$$

This dynamic key, known as *DK*, is divided into three sub-keys, the first two of which (*K1* and *K2*) each have a length of 128 bits, while the third sub-key, known as *Kseed*, and has a length of 256 bits. See figure 3.3

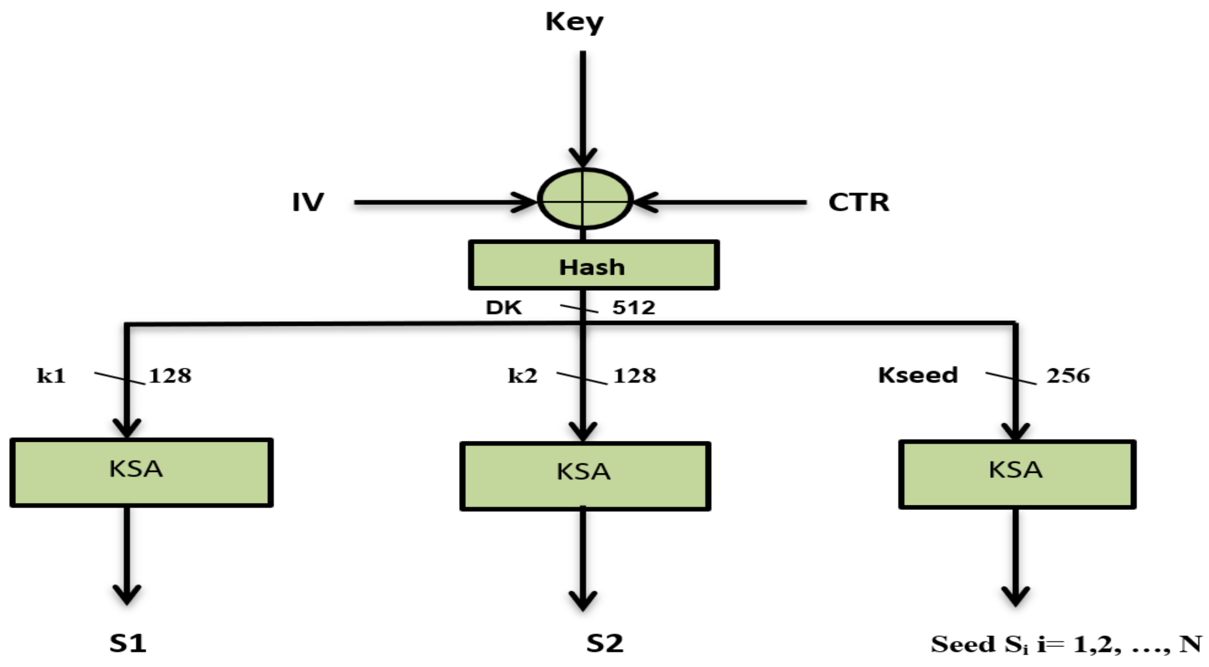


Figure (3.3): The Proposed Dynamic Key Generation

An explanation of each of these sub-keys is provided for your convenience in the following:

1. *K1* is the first substitution sub-key, and it represents the first 128 bits of *DK* that are considered to be the least important. The reason for this is because *K1* is the first subkey in the set. This sub-key is put to use in the process of generating the very first substitution table, which will be Referred to from here on out as *S1*. At this point, you are at liberty to construct dynamic-key dependent replacement tables using whatever method you see fit, and it is your choice which method to choose. For example, the Key Setup Algorithm (KSA) of RC4 was constructed in the same way as described. As a result, it would be possible to develop dynamic replacement tables. At this stage, we also make use of the KSA algorithm, which is used by the RC4 protocol.

2. *K2* is the second replacement sub-key, and it represents the second of the 128 bits of *DK* that are considered to be of the least importance. *K2* comes after *K1*. This sub-key is used to construct the second

substitution table, which is referred to as *S2*, using the same technique as is done with *K1*, which is the KSA for RC4.

3. The first 256 most significant bits (MSB) of *DK* are represented by *KSeed*, which acts as a representation for those bits. This sub-key is used as a hidden seed in combination with any PRNG in order to produce a key stream with a length of *N* words, where the length of each word may vary between 32 and 64 bits. This sub-key is used to generate a key stream with a length of *N* words. As a consequence of this.
4. *KSeed* is the place where one may purchase *N* seeds. Each process will choose one of these created samples to use, and those samples will be generated in a way that is meant to appear random but is really a dynamic simulation.

### 3.2.2 Encryption Method

1-This function is designed to perform a one-round encryption on a given input message, represented by the variable at, using a set of operations.

2-The input is processed by applying bit-level XOR operations with the key, the block counter CTR and the random values generated by the pseudo-random number generator (PRNG) using xorshift64 that applies to the initial vector and the key.

3-The resulting value is stored in another variable.

4-Then, the algorithm applies a Substitution Switch Network (SPN) to this variable using two switch boxes, which are represented by the arrays Sbox1 and Sbox2

5-An SPN consists of eight steps of substitution, where each step operates on a byte of the variable in which the result is saved

6-The S-box sets each byte of this variable to a new value based on a lookup table.

7-The resulting value is sent to the splitmix64 PRNG function and the output is stored again.

8- In general, the algorithm combines PRNG operations, bitwise XOR, S-box, and final XOR operations to perform encryption on the inputs

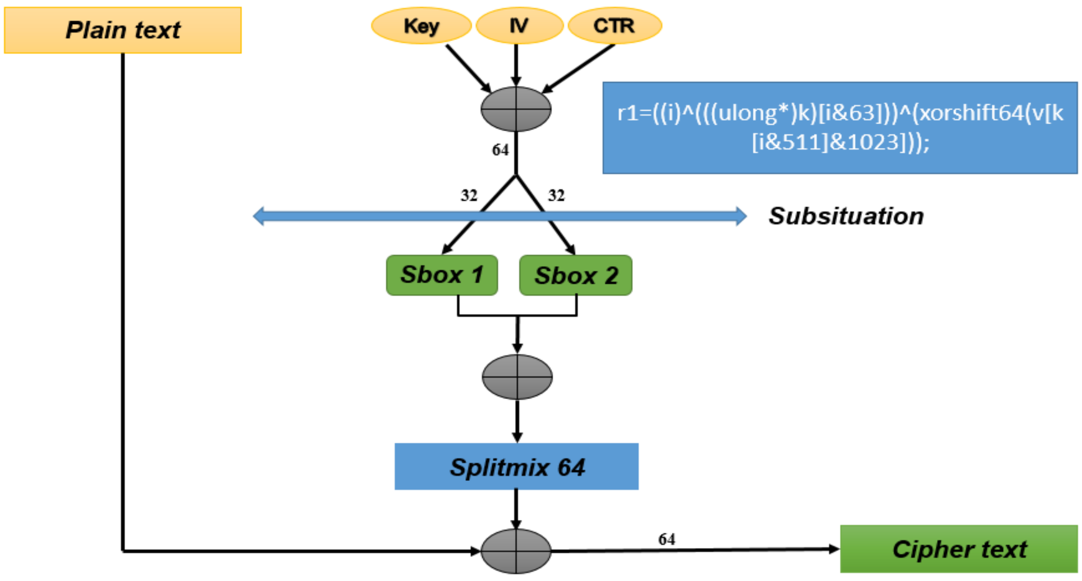


Figure (3.4): Proposed Lightweight Encryption Function

***Algorithm 1*** *The proposed one-round cipher*

***Input:***

- *in: plain block data of size 64-bit*
- *ctr: block counter*
- *Sbox1, Sbox2: arrays of size 256 bytes representing substitution boxes*
- *key: array of size 512 bytes*
- *v: array of initial vector of size 1024*

***Output:***

- *R1: encrypted block of size 64 bit*

```
Start
1. Let R1 be the result of a complex operation:
  - R1 = ((ctr) ^ (((ulong*) key) [ctr & 63])) ^ (xorshift64(v[key[ctr & 511] & 1023]]))

2. Get a pointer to the bytes of R1 and call it vv.
```

```
3. Apply Sbox1 and Sbox2 transformations to vv[0]:
  - vv[0] = Sbox1[vv[0]] + Sbox2[vv[7]]

4. Apply Sbox1 and Sbox2 transformations to vv[1]:
  - vv[1] = Sbox1[vv[1]] + Sbox2[vv[6]]

5. Apply Sbox1 and Sbox2 transformations to vv[2]:
  - vv[2] = Sbox1[vv[2]] + Sbox2[vv[5]]

6. Apply Sbox1 and Sbox2 transformations to vv[3]:
  - vv[3] = Sbox1[vv[3]] + Sbox2[vv[4]]

7. Apply Sbox1 and Sbox2 transformations to vv[4]:
  - vv[4] = Sbox1[vv[4]] + Sbox2[vv[3]]

8. Apply Sbox1 and Sbox2 transformations to vv[5]:
  - vv[5] = Sbox1[vv[5]] + Sbox2[vv[2]]

9. Apply Sbox1 and Sbox2 transformations to vv[6]:
  - vv[6] = Sbox1[vv[6]] + Sbox2[vv[1]]

10. Apply Sbox1 and Sbox2 transformations to vv[7]:
   - vv[7] = Sbox1[vv[7]] + Sbox2[vv[0]]

11. Apply splitmix64 function to R1:
   - R1 = splitmix64(R1)

12. Perform a bitwise XOR operation between in and R1, and assign the result to
R1:
   - R1 = in ^ R1

13. Return the final value of R1.

End
```

***Algorithm 2** : xorshift64*

```
Input: seed (64-bit unsigned integer)
Output: pseudo-random number (64-bit unsigned integer)

1. Initialize state variable: state = seed

2. Define constants:
   - a = 21 (shift value for left rotation)
   - b = 35 (shift value for left rotation)
   - c = 4  (shift value for right rotation)

3. Generate a pseudo-random number:
   - Left shift state by 'a' bits: state = state << a
   - XOR the result with the state shifted left by 'b' bits: state = state ^
(state << b)
   - XOR the result with the state shifted right by 'c' bits: state = state ^
(state >> c)

4. Return the current state as the pseudo-random number.

End
```

### 3.2.2.1 Splitmix64

Among the many advantages of the splitmix64 PRNG are its fast execution time and simple implementation. The"Splitmix64" algorithm, which is shown Algorithm 3, provides a fast split able PRNG. Both the input and output of this function are of size 64 bits. Furthermore, because the generated sequences of pseudo-random values may be anticipated, it is not advised for use in cryptography or security applications. This is because it only takes two successive outputs to rebuild the system's internal state, and the mixing functions are easily invertible. The provided method employs Splitmix-64 with varying seeds and a non-linear operation called confusion. Non-invertible binary diffusion is used to perform the non-linear operations of substitution and mixing. Splitmix-64 uses a well-crafted linear transformation based on shift and rotation in this case. This guarantees a substantial reduction in latency and a substantial measure of randomness in the use of associated resources.

**Algorithm 3 :** Splitmix64

```
Input: seed (64-bit unsigned integer)
Output: pseudo-random number (64-bit unsigned integer)

1. Initialize a state variable: state = seed

2. Define a constant multiplier: multiplier = 2685821657736338717

3. Generate a pseudo-random number:
   - Update the state: state = state + multiplier
   - Perform an integer multiplication of the state by the multiplier
   - XOR the result with the state
   - Right shift the result by 32 bits

4. Return the resulting 64-bit pseudo-random number.

End
```

### 3.2.2.2 RC4

Algorithm 4, performs the key setup process of RC4, initializing the S-box (*sc*) using the key provided. The *S-box* is initialized by swapping the values based on the key, ensuring the key's influence is spread throughout the *S-box*. The resulting initialized S-box (*sc*) can be used for encryption or decryption using the RC4 algorithm.

**Algorithm 4 :** RC4

```
Input: Key (variable-length), Plaintext (variable-length)
Output: Ciphertext (same length as plaintext)

1. Initialize the S-box (256-byte array) and a key-based permutation of the S-box:
   - Create an S-box as an array of integers from 0 to 255 (S[0] to S[255]).
   - Generate a key-based permutation of the S-box by mixing it with the key using a key-scheduling algorithm.

2. Initialize two pointers, i and j, to 0.

3. Encryption:
   - For each byte in the plaintext:
     - Increment i and j.
     - Swap S[i] and S[j].
     - Calculate the pseudo-random key byte (keystream byte) as S[(S[i] + S[j]) % 256].
     - XOR the plaintext byte with the keystream byte to produce the ciphertext byte.

4. Repeat the encryption process for the entire plaintext.

5. Return the ciphertext.

End
```

**Algorithm 5:** Dynamic key Generation algorithm

```
Input: seed (uint64_t) - An initial seed value
Output: DK (array of uint64_t) - The dynamic key array

1. Initialize an array named DK of size 512 elements, each of type uint64_t, to store the dynamic key values.

2. For i ranging from 0 to 511:
   a. Calculate DK[i] by applying the XORShift64 algorithm to the sum of seed and i.

3. Increment the value of seed by 512 to prepare for the next set of dynamic key values.

4. Allocate memory for an array named iv of size n elements, each of type uint64_t, to store initialization vectors.

5. Allocate memory for an array named Sbox of size 256 elements, each of type uchar, to store the S-box values.

6. For j ranging from 0 to n-1:
   a. Compute the value of iv[j] using the SplitMix64 algorithm with the sum of seed and j.

7. Increment the value of seed by n to prepare for the next set of initialization vectors.

8. Invoke the RC4KeySetup function, passing the address of DK[256], Sbox, and the value 256 as arguments to set up the RC4 cipher.

End
```

### 3.2.3 Proposed Parallel Authentication Algorithm

Algorithm 7 accepts the following input parameters: a 64-bit integer array denoting plain data, a 64-bit integer array denoting *v*, two 8-

bit unsigned character arrays denoting *Sbox1* and *Sbox1*, a character array denoting *DK*, four 64-bit integer pointers denoting *lmac*, an integer size, an integer *numtasks*, and an integer *id* denoting the index of a parallel thread. It takes the information you provide and returns a message authentication code (MAC). Four 64-bit integer references are set to 0 by the procedure. The thread block size is then calculated by dividing the size data array by the number of blocks, *numtasks*. Each thread's beginning and ending indexes are then computed. The method then loops four times through the data blocks for each thread. The encrypt function is called with the necessary arguments to create the four 64-bit integers *r1*, *r2*, *r3*, and *r4* for each block. Then, we XOR these four numbers together to produce a temporary 64-bit integer t. After putting t's eight bytes through *Sbox1*, t is updated with the new values. Then, we XOR the resultant four numbers with the corresponding entry of the *lmac* array, and finally, we run the splitmix64 function using t as the input to build a new 64-bit integer. The MAC for the input data is stored in the resultant *lmac* array. The parallel

Authentication function is shown in Figure 3.1. All threads' *lmac* arrays are merged into a single *all_mac* array through an asynchronous collecting process. The *all_mac* array is compacted to the *final_mac* array by authentication procedures performed in the master thread, thread 0. Without the use of encryption, the implemented authentication processes are identical to those of thread-level authentication.

**Algorithm 7:** Parallel MAC Algorithm (PAA-256)

```
Input:
data: array of uint64_t data blocks
v:  initial vector of type uint64_t
size: size of data array
numtasks: number of parallel tasks
id: Index of the current task
Sbox1: array of 256 bytes used for S-box substitution
Sbox2: array of 256 bytes used for S-box substitution
DK: array of bytes used for encryption key
Master: master thread that set to 0
lmac: array of four uint64_t values used to
     store the local MAC of a thread
all mac: array of size 4*numtasks of type uint64_t
Output:
    final_mac: array of size 256 bits
Start
1. Set lmac[0], lmac[1], lmac[2], and lmac[3] to 0.
   threadbl = ceil (size / numtasks)
   start = id * threadbl
   end = start + (threadbl- 1)
2. for (int i= start; i < end; i+=4)
{
     // Encrypt each data block using the encrypt
       function:
      r1 = encrypt(data[i], i, Sbox1, Sbox2, DK, v)
      r2 = encrypt(data[i+1], i+1, Sbox1, Sbox2, DK, v)
      r3 = encrypt(data[i+2], i+2, Sbox1, Sbox2, DK, v)
      r4 = encrypt(data[i+3], i+3, Sbox1, Sbox2, DK, v)
      t = r1 ^ r2 ^ r3 ^ r4
      uchar *rr=(uchar*)&t;
      for (j = 0; j < 8; j++)
           rr[j]=Sbox1[rr[j]];
      t = splitmix64(t)
      lmac[0] ^= r1 ^ t
      lmac[1] ^= r2 ^ t
      lmac[2] ^= r3 ^ t
    lmac[3] ^= r4 ^ t
}
3.Applying an asynchronous gathering operation for all mac values from all threads
  MPI_Igather (lmac, 4, MPI_INT64_T, all_mac, 4,   MPI_INT64_T, MASTER, MPI_COMM_WORLD)
4.if (id==MASTER)
    Apply the authentication steps for the all_mac array as in the steps b, c and
    d  while storing the results in the final_mac array
5.Return final_mac array
End
```

## 3.3 Summary

This chapter describes the complexities of the Proposed Parallel Message Authentication algorithm, its underlying structure, and the encryption and authentication techniques used for messages.

# CHAPTER FOUR

# EXPERIMENTAL RESULT

## 4.1 Introduction

This chapter presents a comprehensive examination of the outcomes derived from our proposed methodology, focusing on the sensitivity tests encompassing the pdf test, histogram analysis, entropy measurement, correlation assessment, and difference evaluation. Moreover, an in-depth exploration of the performance evaluation is undertaken, comprising three key criteria, namely execution time, speedup, and throughput. Finally, a comparative analysis of the results is conducted on two distinct CPUs exhibiting contrasting specifications. The first CPU specification entails the utilization of an Intel(R) Core(TM) i7-10510U CPU clocked at 1.80GHz, whereas the second CPU specification employs an Intel(R) i7-7700HQ CPU running at 2.80GHz.

## 4.2 Security Analysis

The suggested authentication technique is tested against common attacks including statistical, linear, differential, and brute-force attacks to make sure it can withstand them. In this part, extensive tests are conducted to prove that our suggested authentication method is safe and effective. While the proposed technique is applicable to any kind of data, findings for multimedia items below.

### 4.2.1 Statistical Analysis Tests

Authentication and encryption against statistical attacks need randomness and homogeneity. Evaluation of randomness, as well as tests for statistical integrity such as examination of the PDF, histogram, entropy, and correlation between the original and encrypted messages, are performed. Table (4.1) shows the average of security measurement for tested images.

### 4.2.1.1 Histogram Analysis

The histogram of the encrypted message has to be normally distributed so that the uniformity of the proposed one-round encryption utilized in the authentication method may be tested. This means that each symbol appears at a rate that is roughly proportional to the total number of symbols. As little as possible while yet conveying the intended meaning of the message. Figure (4.1) show a histogram comparing the plain-text and Figure (4.2) cipher-text versions of a 512-by-512 for Lena image and Figure (4.3) show a histogram comparing the plain-text and Figure (4.4) cipher-text versions of a 512-by-512 for Boat image. It demonstrates that encrypted messages follow a distribution very similar to the conventional histogram. Our encrypted message has a histogram that is somewhat close to (512x512)/256, or 1024.

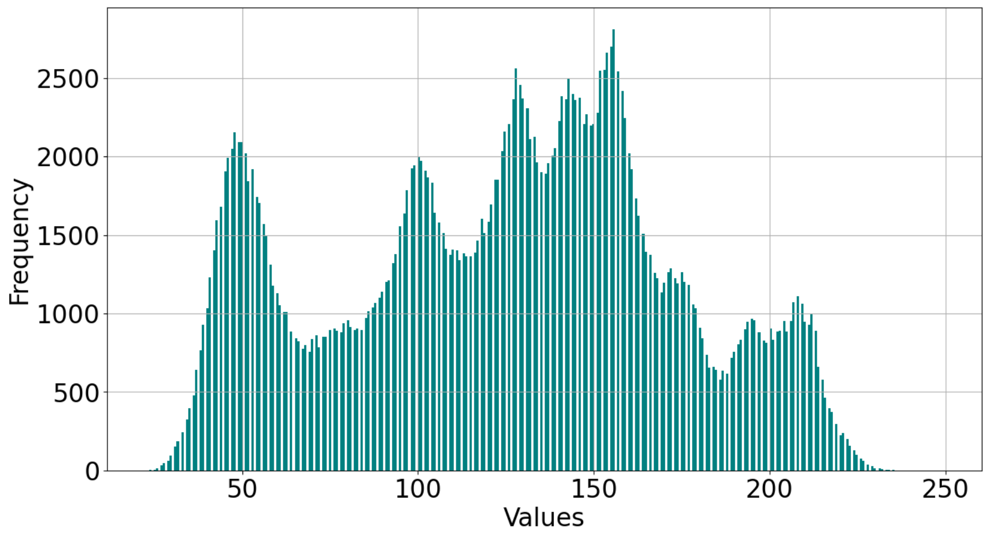


Figure (4.1): Histogram of Plain Text for Lena Image

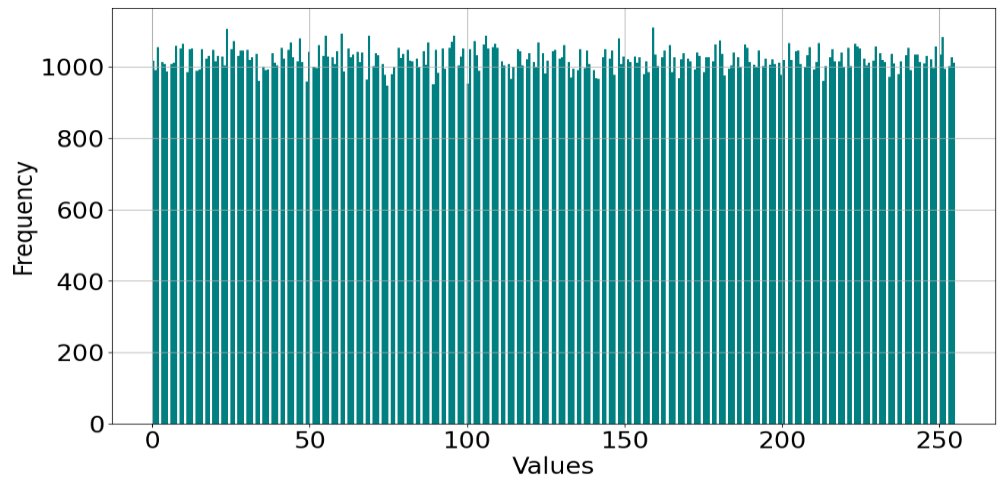


Figure (4.2): Histogram of Cipher Text for Lena Image

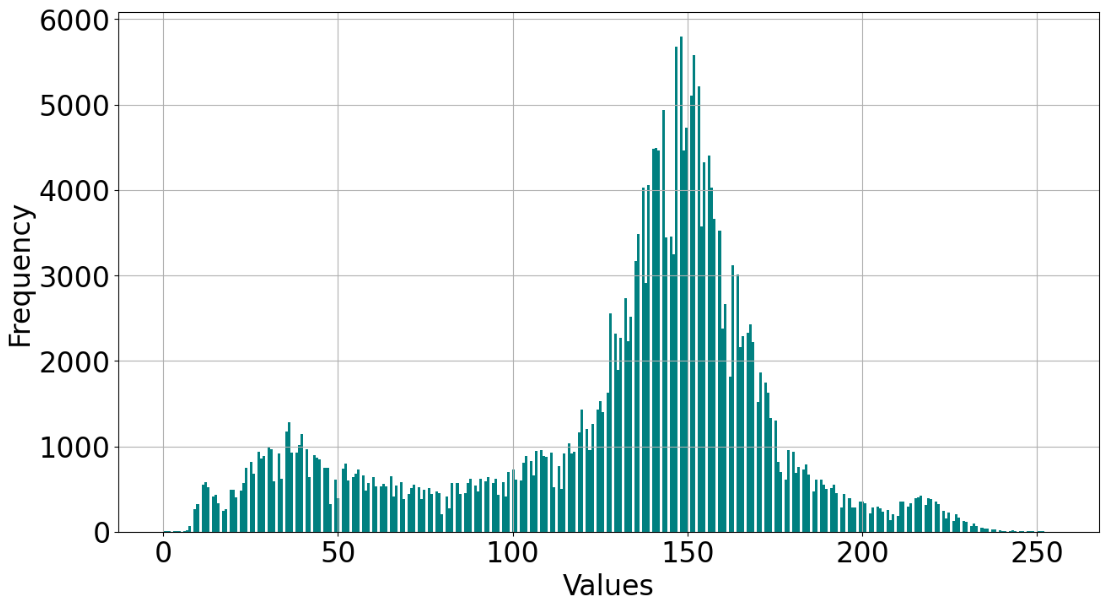


Figure (4.3): Histogram of Plain Text for Boat Image

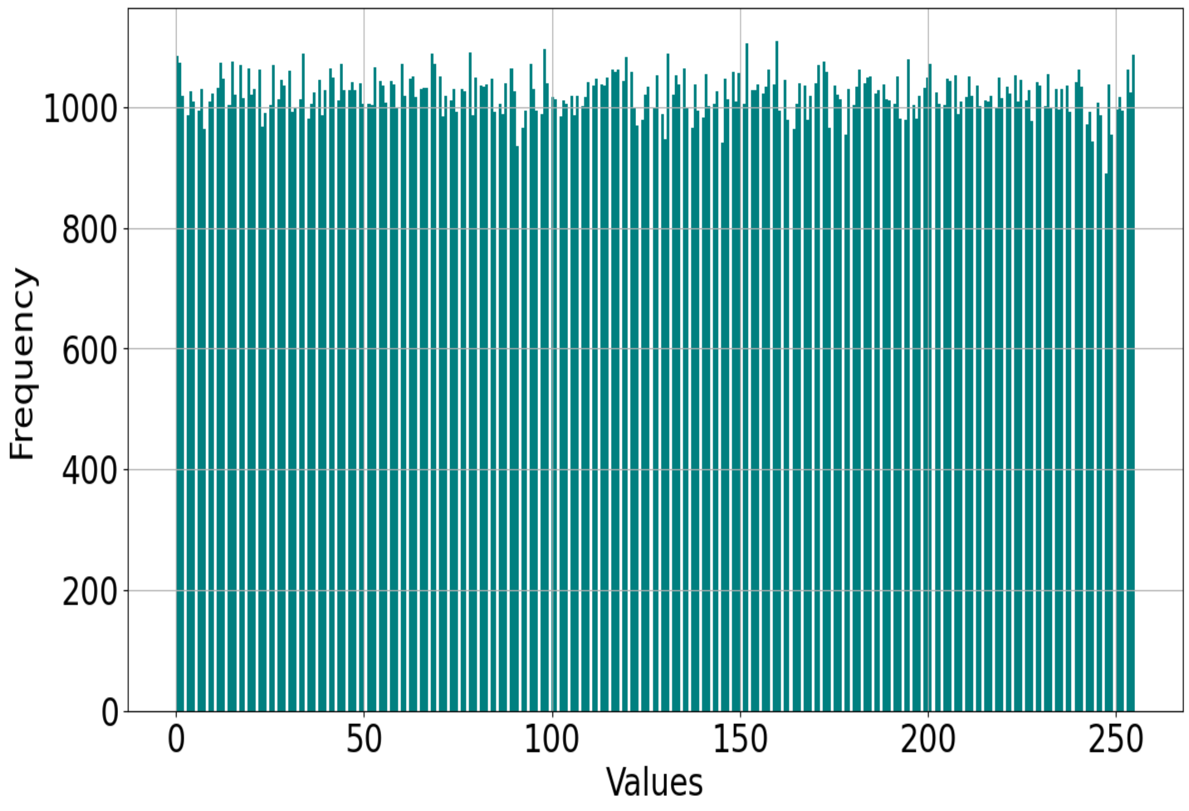


Figure (4.4): Histogram of Cipher Text for Boat Image

### 4.2.1.2 Uniform Security Test

The PDF (probability density function) of the encrypted message must be uniform in order to pass this test, since it is the test with the most significance. Each symbol in the ciphertext has a chance of occurrence that is very near to *1/n*, where n is the total number of symbols used in the cipher. There are two versions of the PDF of the encrypted message and the original one shown in Figures (4.5) and (4.6) for Lena image and two versions of the PDF of the encrypted message and the original one shown in Figures (4.7) and (4.8) for Boat image. All of the ciphertext symbols have a probability density function (PDF) with a value of 0.039 (1/256 = $3.9\times10^{3}$), which is close to the uniform distribution. Figure (4.9) explains the recurrence of the original message and Figure (4.10) cipher text for Lena image and Figure (4.11) explains the recurrence of the original message and Figure (4.12) cipher text for Boat image.

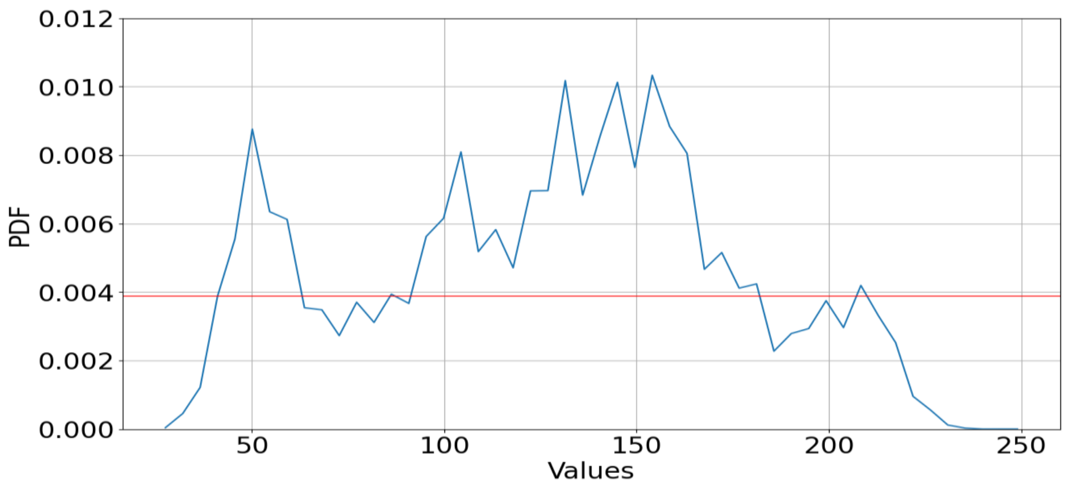


Figure (4.5): PDF of Plain Text for Lena Image

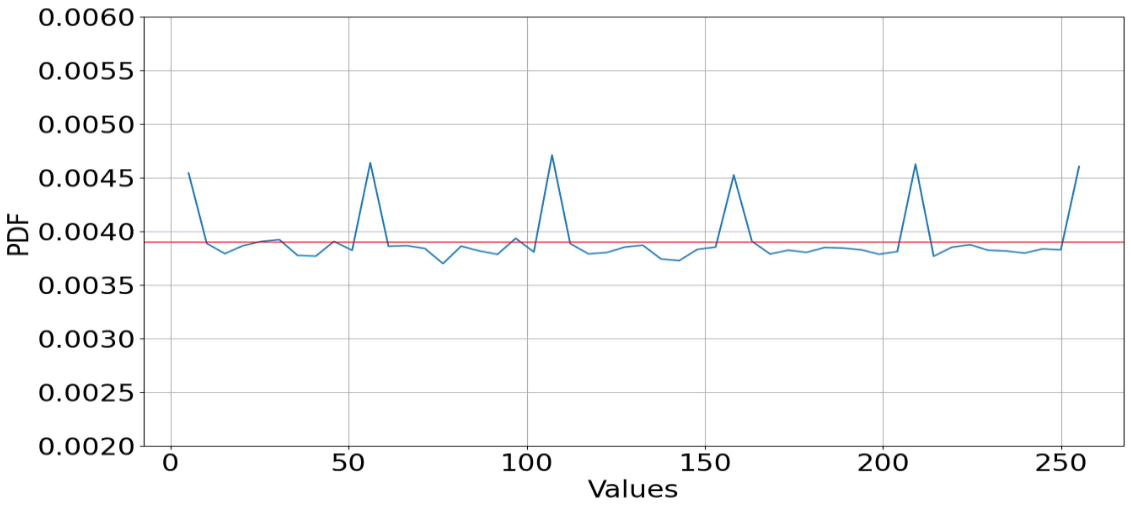


Figure (4.6): PDF of Cipher Text for Lena Image

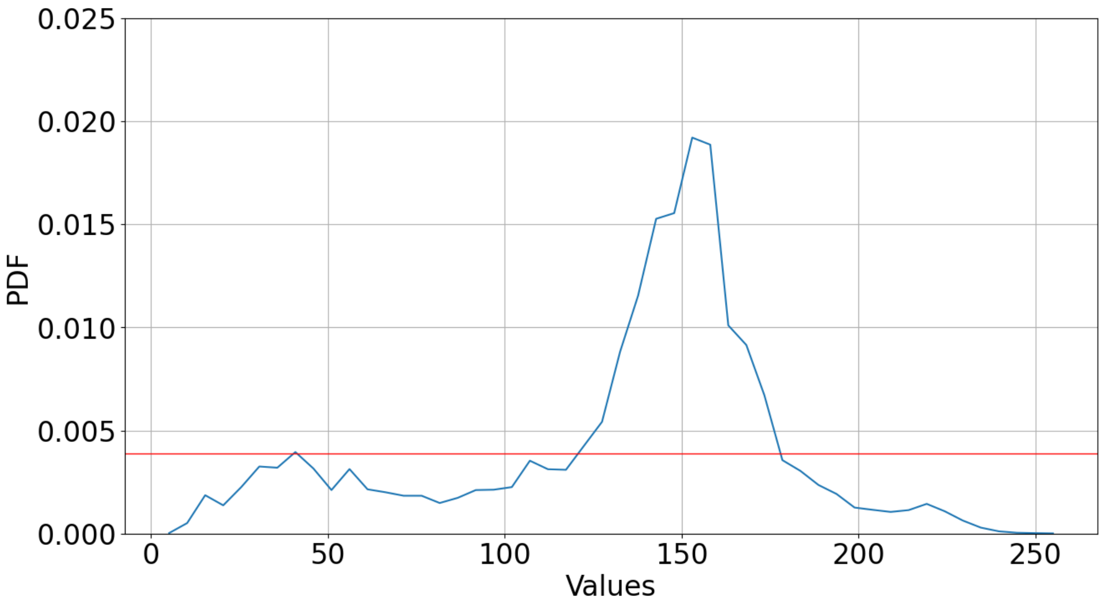


Figure (4.7): PDF of Plain Text for Boat Image

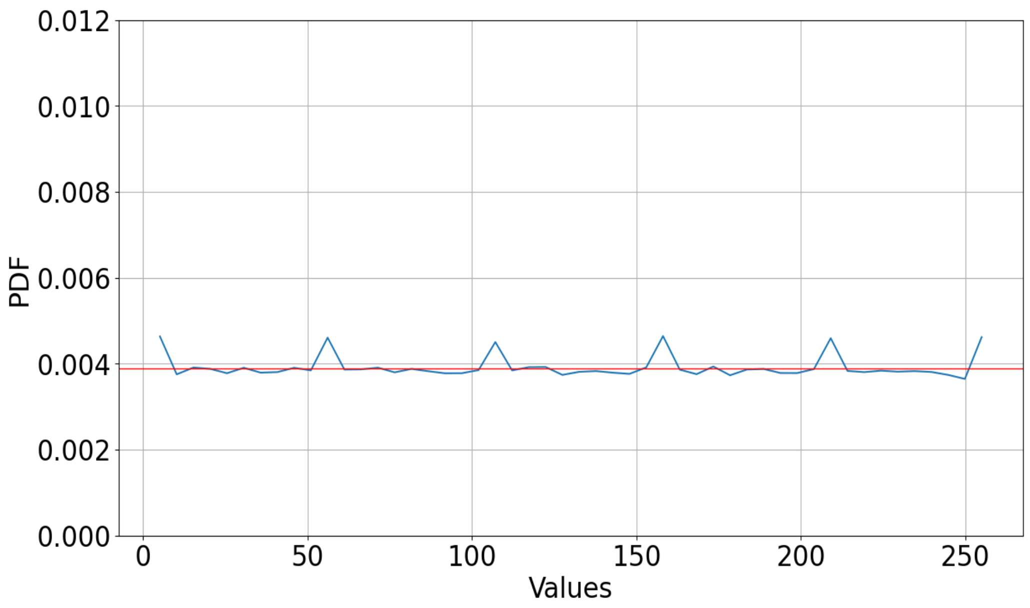


Figure (4.8): PDF of Cipher Text for Boat Image

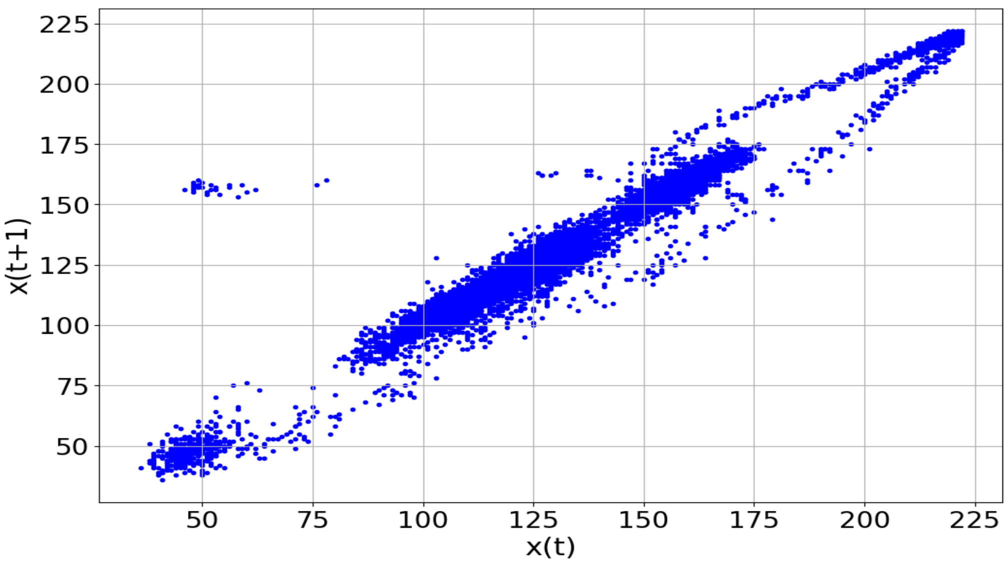


Figure (4.9): The Recurrence of the Original Message for Lena Image

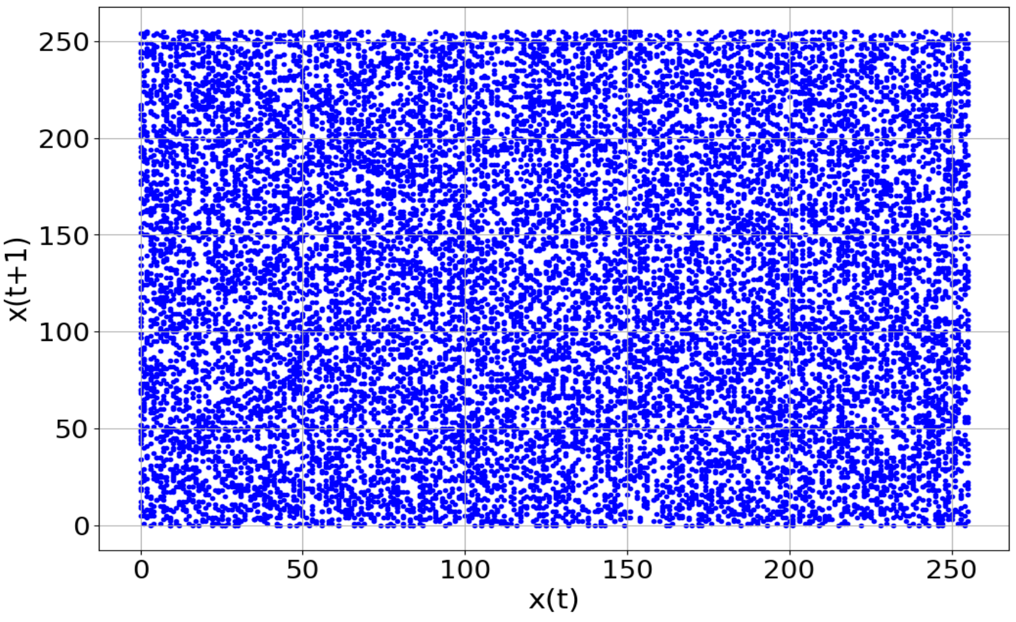


Figure (4.10): The Recurrence of Cipher Text for Lena Image

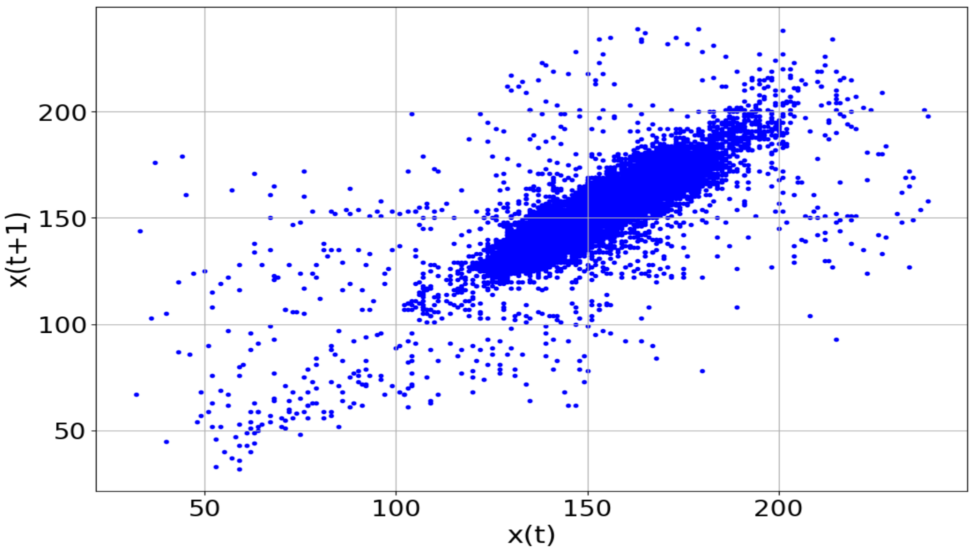


Figure (4.11): The Recurrence of Plain Text for Boat Image

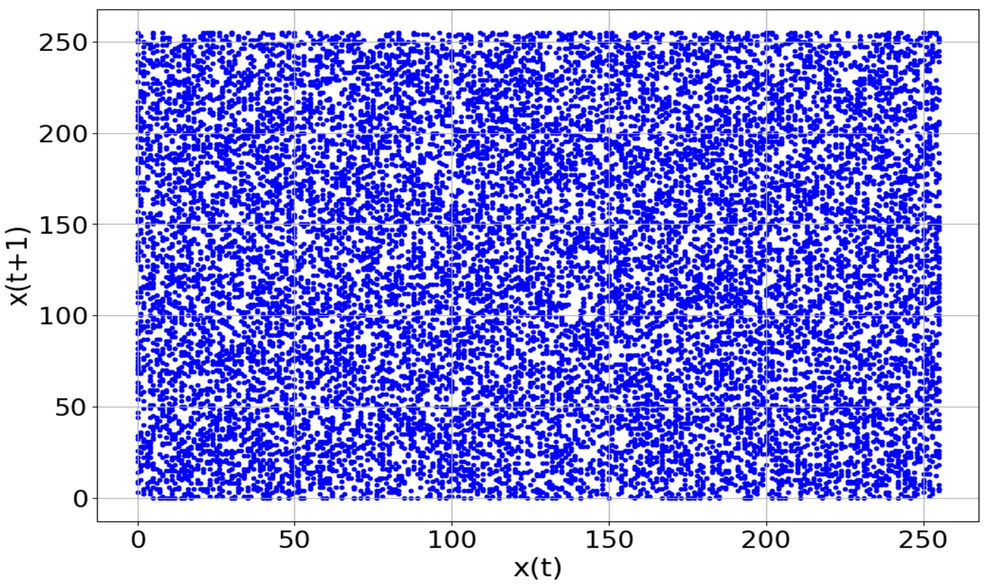


Figure (4.12): The Recurrence of Cipher Text for Boat Image

### 4.2.1.3 Entropy Analysis

This section is expounded upon in accordance with the guidelines outlined in Section 2.7.1.3 of Chapter Two. Figure (4.13), shows the results of a submatrix-level analysis of the entropy of an encrypted message with a random dynamic key, using a matrix of dimensions 16 x 16 (256 elements). This action was taken so that the project might be finished. In Figure (4.14), the collected statistics demonstrate that the cipher messages produced have an entropy equal to 5, the target number for Lena image and Figure (4.15), shows the results of plain text for Boat image and in Figure (4.16) ciphertext for Boat image, the collected statistics demonstrate that the cipher messages produced have an entropy equal to 5, the target number. As a result, the proposed encryption technique is secure enough to withstand any entropy attack. Based on the findings, the cipher message that was generated had optimally uniform entropy. Against any entropy attack, the suggested encryption technique provides enough security.

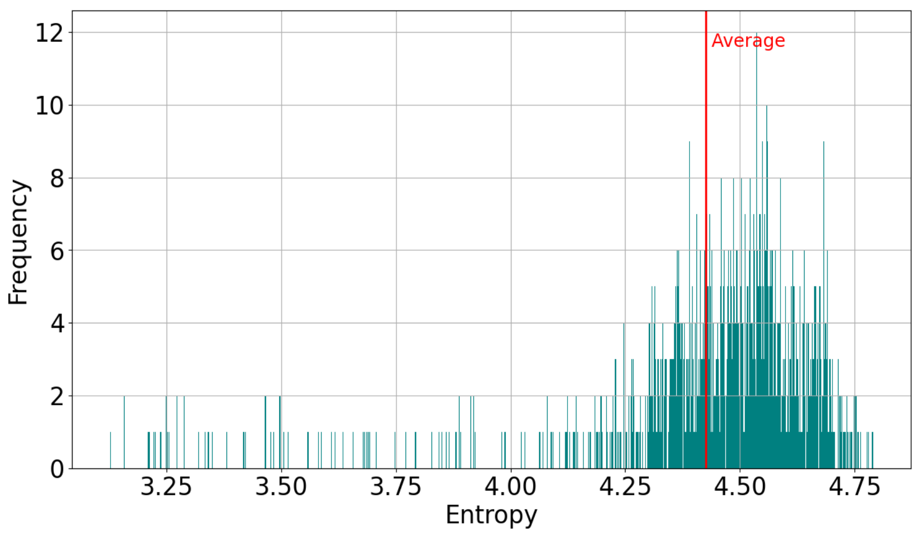


Figure (4.13): Entropy of Plain Text for Lena Image

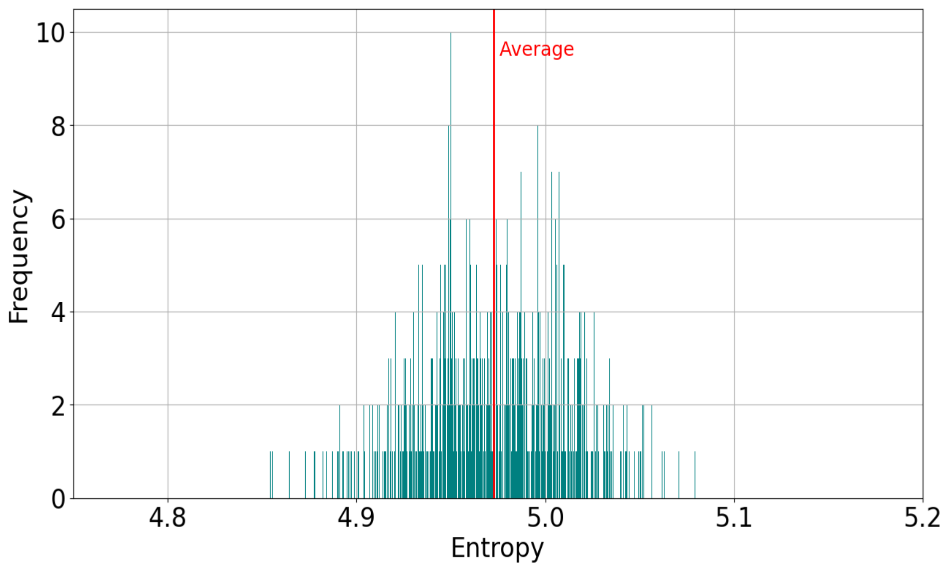


Figure (4.14): Entropy of Cipher Text for Lena Image

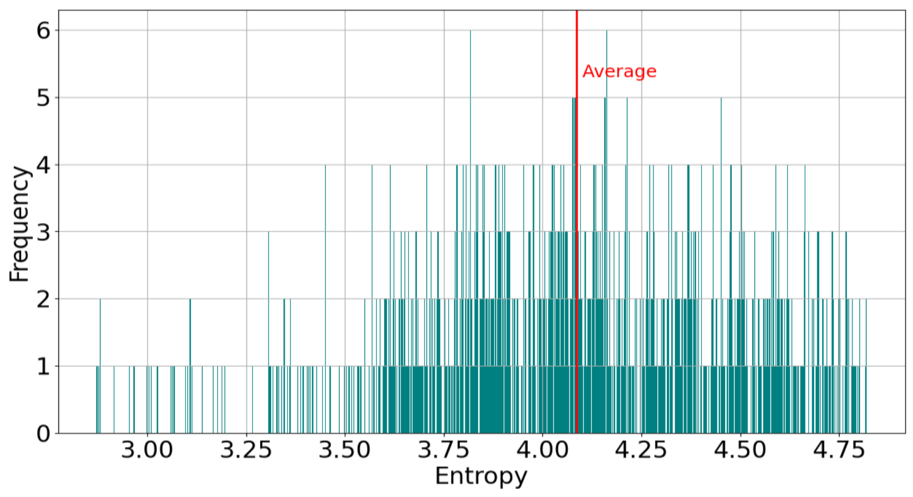


Figure (4.15): Entropy of Plain Text for Boat Image

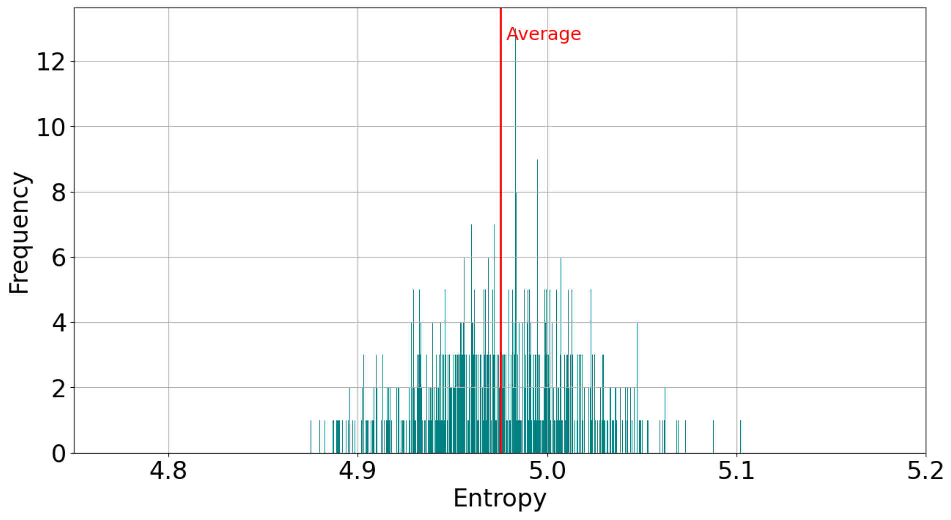


Figure (4.16): Entropy of Cipher Text for Boat Image

Table (4.1): The Average of Security Measurement for Tested Images

| **Image name** | **The average of security measurement** | | |
|---|---|---|---|
| | **Histogram** | **PDF** | **Entropy** |
| **Lena plain** | **748.98** | **0.00392** | **4.424** |
| **Lena cipher** | **748.98** | **0.00392** | **4.972** |
| **Boat plain** | **748.98** | **0.00392** | **4.085** |
| **Boat cipher** | **748.98** | **0.00392** | **4.975** |

### 4.2.1.4 Randomness Test Using PractRand

The suggested authentication technique and the one-round cypher were both tested using a total of 1000 seeds using PractRand, and both passed with flying colors. Already said up above. Actually, a 512-by-512 pixel message is the sole one used. Each part of the message is set to zero at the outset, and the key is generated only once. Furthermore, no additional variables are initialized more than once. In order to decipher the resulting message, one of the most challenging statistical tests, PractRand, was used. These analyses will help determine whether the resulting keystream satisfies the required randomization and uniformity criteria. In addition, the Practrand test suite seeks for certain kinds of non-randomness that may be missed by standard statistical testing. Gap tests, birthdate spacing tests, and other statistical analyses are only some of the methods used by PractRand. Quantitative measurements of statistical quality are also provided, such as the amount of output in "*TB*" (terabytes) required to identify a certain degree of non-randomness.

#### 4.2.1.5 Correlation

In Chapter Two, a comprehensive explanation of the concept of independence was provided, along with a detailed exposition of the methodologies employed for calculating its averages, as presented in Section 2.7.1.3. Figure (4.17) shows the results of a correlation test between the original and encrypted communications, which were encrypted using a technique in which one thousand random keys were used sequentially. Results point to a low value for the correlation coefficient across messages, which suggests that the ciphertext is very random and independent.

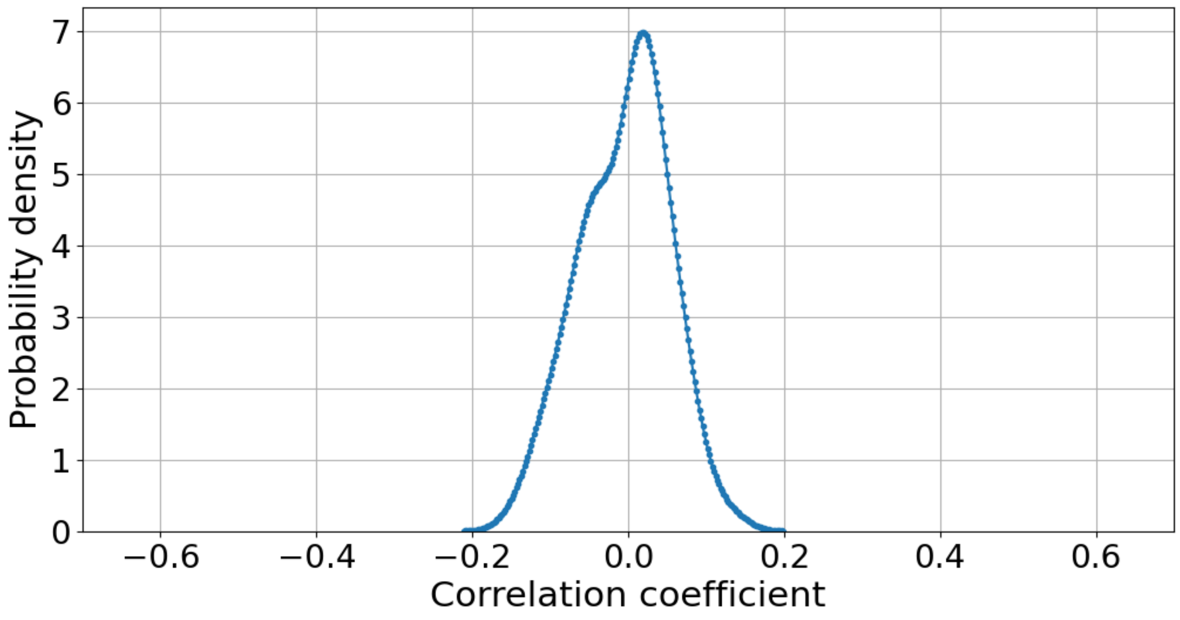


Figure (4.17): Distribution of the Correlation Coefficient between the Plain and the Encrypted Message for 1,000 Randomly Generated Keys.

### 4.2.2 Sensitivity test

In Chapter Two, an extensive analysis of sensitivity testing was conducted, specifically within the context of section 2.7.2. The proposed authentication method becomes more sensitive as differences increase. For *Nb* =256 and similarly for greater values of *Nb*, Figure (4.18) shows that 1000 MAC values for the same message may be distinguished from one another by altering a single bit of the key in a completely random fashion. As can be seen from the findings collected, there is a change of at least *Nb2/*bits (or 50% of MAC bits). The suggested MAC, on the other hand, guaranteed a high degree of unpredictability in the face of statistical assaults.

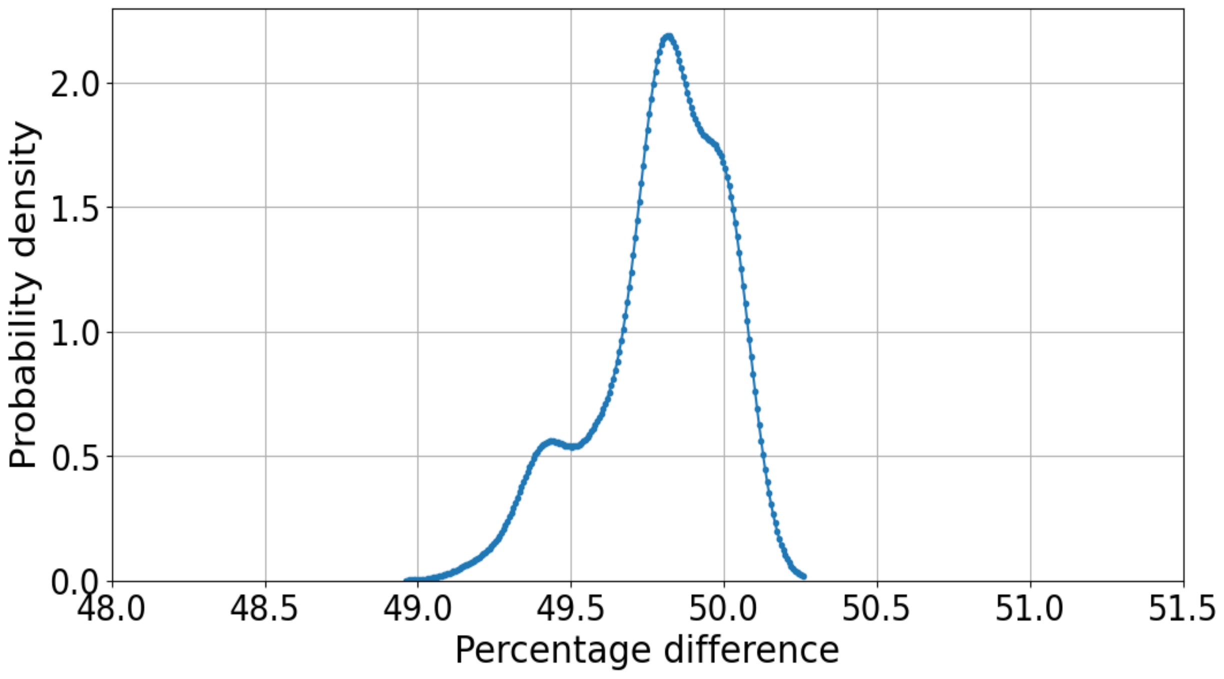


Figure (4.18): The Percentage Difference of the Sensitivity Test between 1000 MAC Values Generated by Changing Randomly One bit In the Key for the Same Message

## 4.3 Performance Evaluation

### 4.3.1 Experiment Setting

In this section, an exposition of the findings pertaining to two distinct processing units is provided: the first CPU specification entails the utilization of an Intel(R) Core(TM) i7-10510U CPU clocked at 1.80GHz, whereas the second CPU specification employs an Intel(R) i7-7700HQ CPU running at 2.80GHz. A comprehensive overview of the average execution time and throughput for all message sizes is presented in Table 4.2 for CPU1, while Table 4.3 encapsulates the corresponding results for CPU2.

### 4.3.2 Performance Results

This subsection details the throughput, execution time, and speedup achieved by the proposed parallel authentication technique. The data message is compressed to a 256-bit MAC value in a series of stages in the proposed technique. The approach relies on a lightweight encryption algorithm that, with a single round of processing, generates a 64-bit block. In this subsection, however, how well the suggested authentication mechanism works with various lightweight ciphers must be assessed. A MAC value is calculated by comparing the Speck encryption algorithm to the one being proposed. A lightweight speck with a block size and key length of 128 is utilized for the comparison. Algorithm 4's call of speck may be repeated twice to get four 64-bit encrypted blocks. The present study assesses the performance of the proposed parallel authentication and Speck algorithms, employing distinct central processing units (CPUs), namely CPU1 with Intel multicore (R) i7-7700HQ specifications. CPU1 is equipped with four physical cores and eight logical cores, each operating at a frequency of 2.8 GHz. Additionally, the evaluation encompasses CPU2, characterized by an Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz base frequency.

Experiments are run with varying message sizes (8, 6, 32, 64, 128, and 256 megabytes) and thread counts (2, 4, 6, and 8). Parallel execution of both algorithms over several threads is shown in Figures (4.19) and (4.20). The suggested approach has been shown to have faster throughput than the speck-based authentication mechanism. The throughput data are shown in Figures (4.21) and (4.22), which show that the suggested parallel authentication process has higher performance than the alternative speck approach. The speedup is an important indicator of performance in parallel computing. In order to get insight into the acceleration obtained while comparing various speeds, we may use the concept of speedup, which is defined as the ratio of the execution time of a sequential application to the execution time of its parallel counterpart. The efficiency gain compared to the serial approach versus the parallel approach is shown in Figure (4.23). The average acceleration factor between the parallel and serial algorithms is 2.99 in all cases involving multiple threads. The efficiency gain compared to the serial versus parallel approach is shown in Figure (4.24). The average acceleration factor between the parallel and serial algorithms is 2.72 in all cases involving multiple threads.

Parallel implementations of both the proposed authentication technique and the speck-based authentication algorithm are shown to exhibit speedup ratios, as shown in Figure (4.25) and figure (4.26). In this situation, a speedup of 3.27 percent is typical. This means that the suggested technique has a lower overall execution cost than the speck-based authentication algorithm.

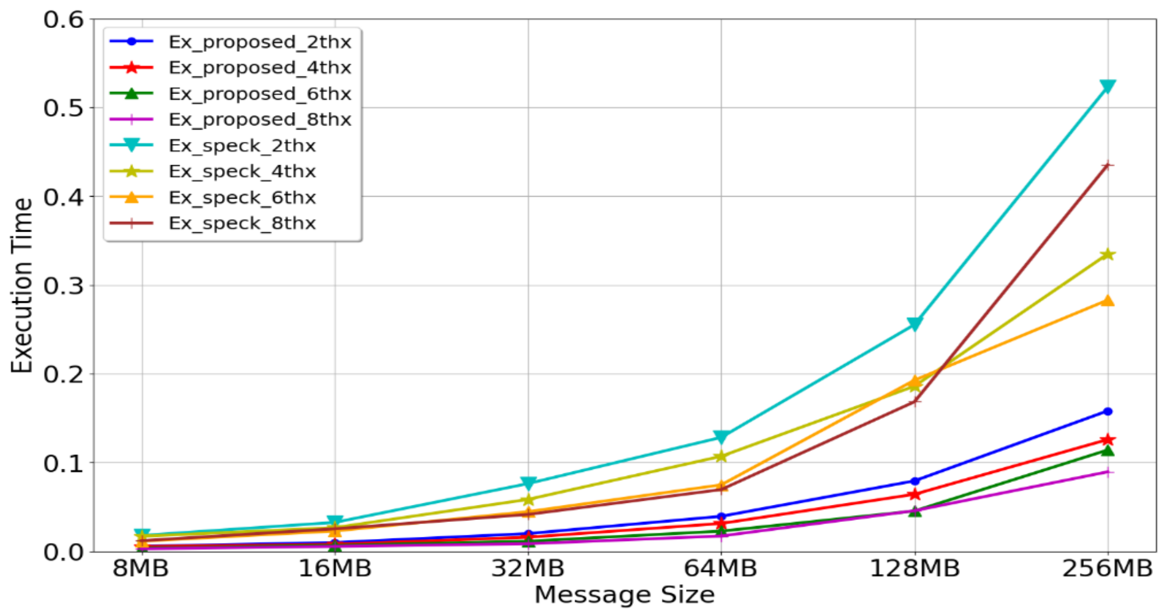


Figure (4.19): Compute Execution Time Over Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

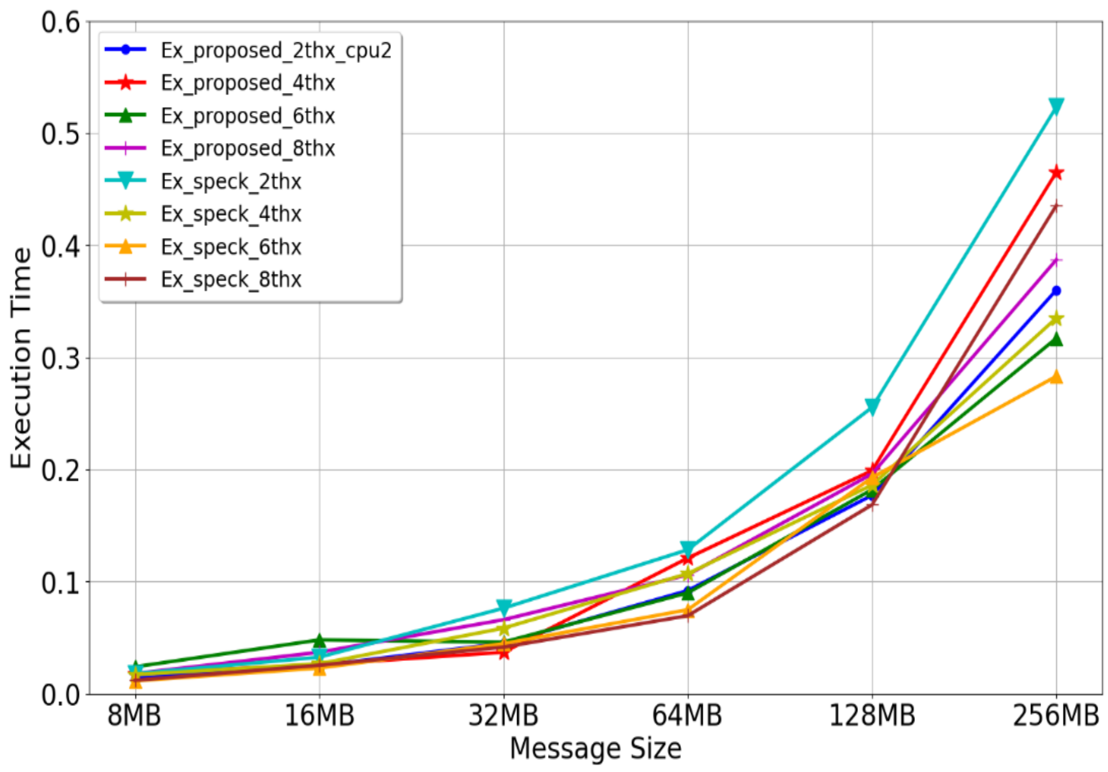


Figure (4.20): Compute Execution Time Over Intel(R) i7-7700HQ CPU @ 2.80GHz

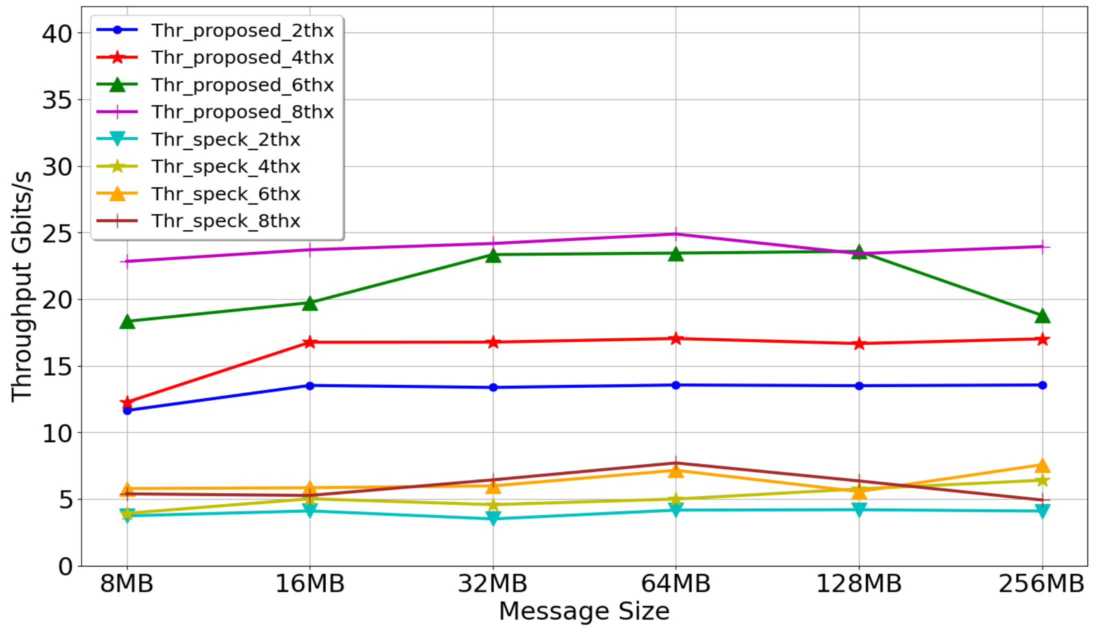


Figure (4.21): Throughput Over Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

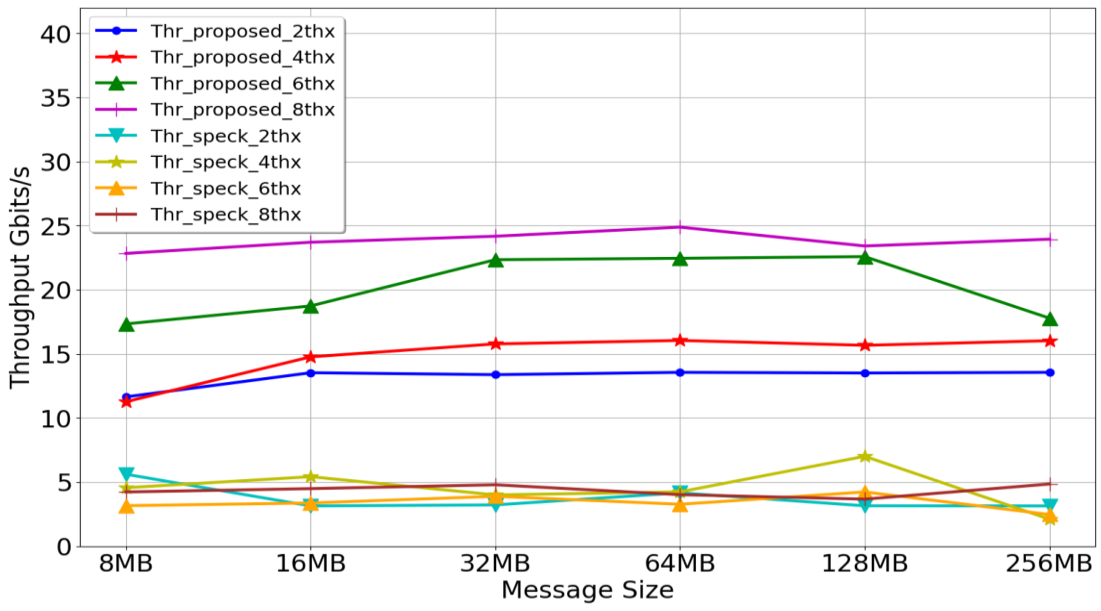


Figure (4.22): Throughput Over Intel(R) i7-7700HQ CPU @ 2.80GHz

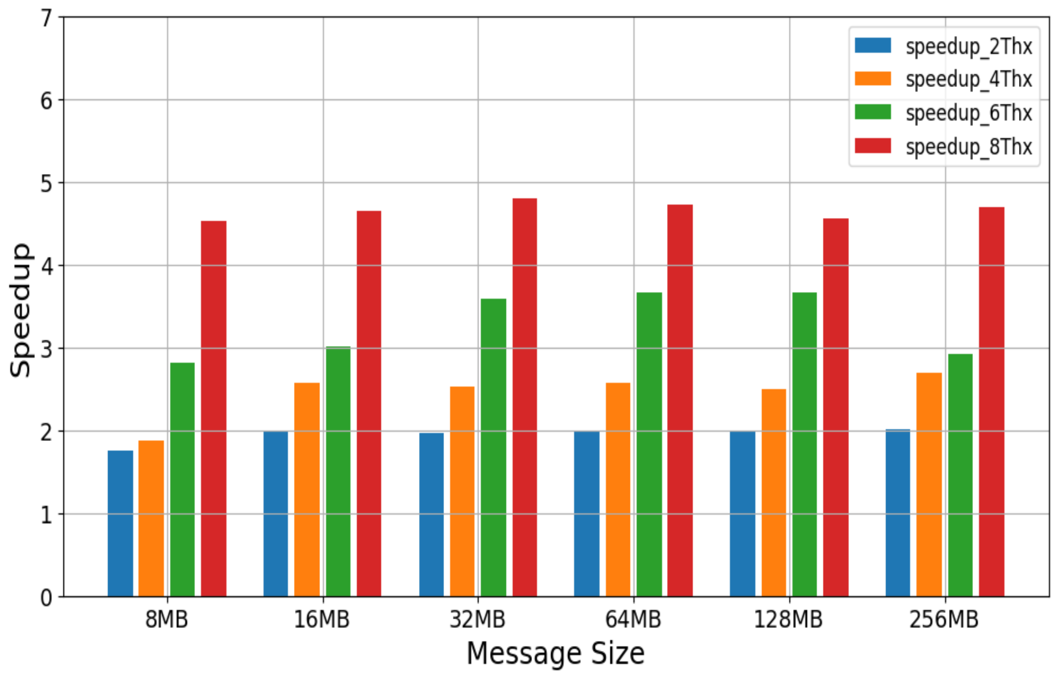


Figure (4.23): Speedup Over Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

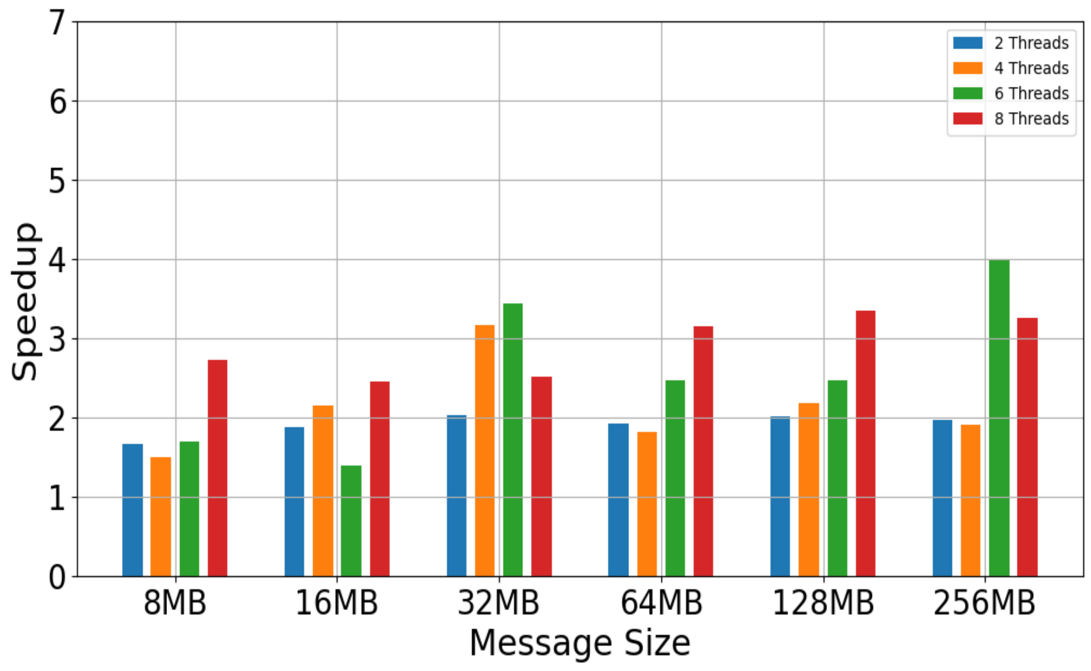


Figure (4.24): Speedup Over Intel(R) i7-7700HQ CPU @ 2.80GHz

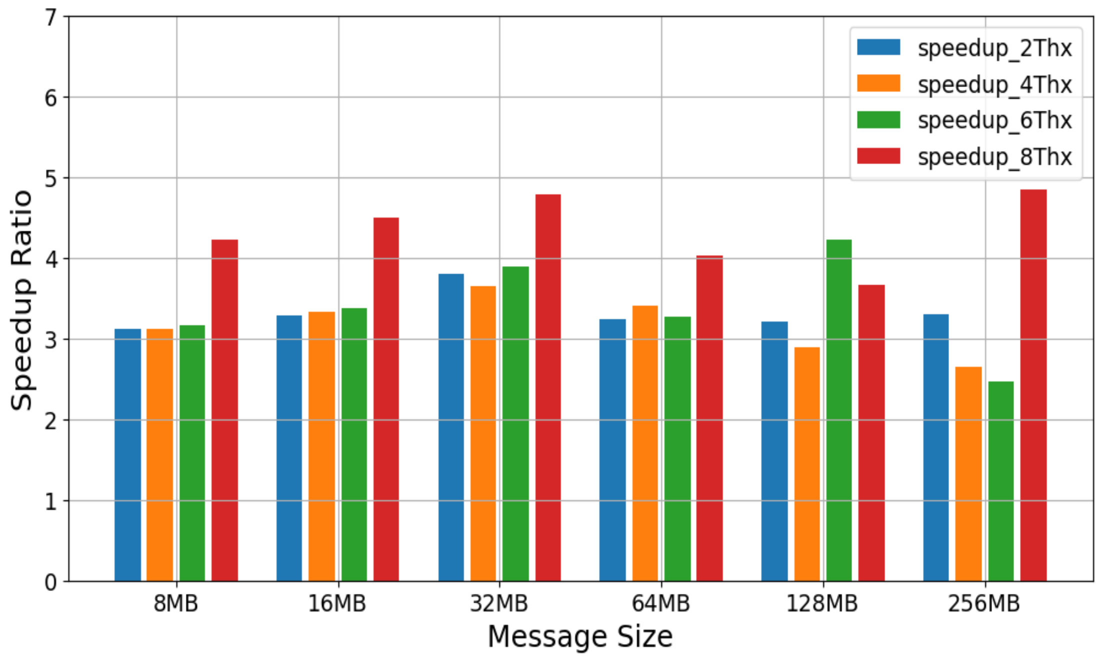


Figure (4.25): Speedup Ratio Over Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

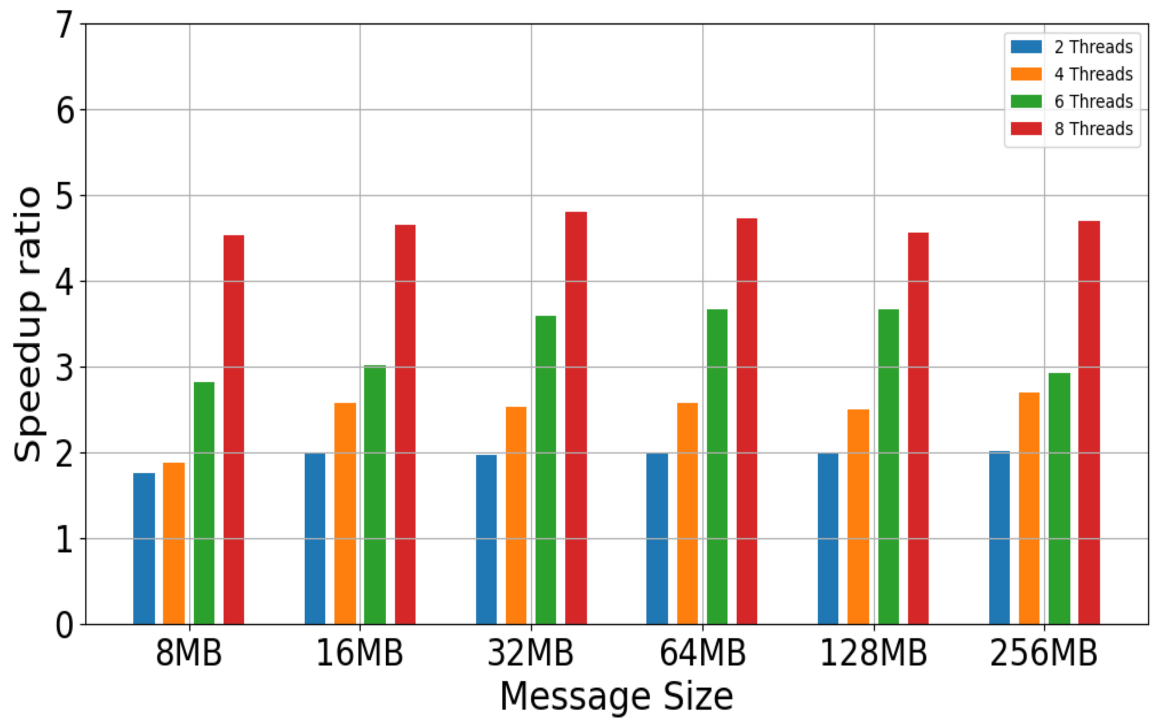


Figure (4.26): Speedup Ratio Over Intel(R) i7-7700HQ CPU @ 2.80GHz

Table (4.2): The Comparison Results Over Intel(R) i7-7700HQ CPU Running at 2.80GHz

| **#threads** | **Average overall message sizes** | | | |
|---|---|---|---|---|
| | **Execution time (s)** | | **Throughput Gbits/s** | |
| | **Proposed** | **Speck Auth.** | **Proposed** | **Speck Auth.** |
| **2** | **0.052** | **0.172** | **13.20** | **3.97** |
| **4** | **0.041** | **0.121** | **16.10** | **5.12** |
| **6** | **0.034** | **0.104** | **21.05** | **6.32** |
| **8** | **0.028** | **0.125** | **23.99** | **5.85** |

Table (4.3): The Comparison Results Over Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

| **#threads** | **Average overall message sizes** | | | |
|---|---|---|---|---|
| | **Execution time (s)** | | **Throughput Gbits/s** | |
| | **Proposed** | **Speck Auth.** | **Proposed** | **Speck Auth.** |
| **2** | **0.053** | **0.144** | **12.32** | **3.75** |
| **4** | **0.033** | **0.119** | **15.01** | **5.08** |
| **6** | **0.030** | **0.101** | **21.00** | **6.17** |
| **8** | **0.026** | **0.115** | **22.97** | **5.66** |

## 4.4 Summary

This chapter describes the results obtained from the chosen Parallel Message Authentication algorithm. Displays the results of sensitivity tests and also displays the results in terms of execution time, throughput, and speedup on two main processing units with different specifications.

# CHAPTER FIVE

# CONCLUSIONS AND FUTURE WORKS

## 5.1 Conclusions

In conclusion, this thesis presents a pioneering framework for parallel message authentication using a novel lightweight cipher. This framework was developed as a result of the study that was conducted. In our approach, the message is first broken up into blocks with a size of 64 bits, and then encryption and authentication procedures are carried out in a single iteration. The current suggested method has been shown to be superior to other approaches, such as the Speck method, which requires a number of rounds and, as a result, has decreased speed and performance in comparison to our method. This has been shown via empirical evaluations. When compared to the parallel speck-based authentication method, the suggested authentication method is about 3.27 times quicker when executed on a multicore central processing unit (CPU). When compared to its sequential version, the suggested approach, when carried out in parallel, demonstrates an improvement that is equivalent to a factor of 2.99.

In addition, both the suggested authentication method and its encryption methodology have been successful in passing the rigorous randomization test known as PractRand. This accomplishment highlights the resilience and dependability of the technique that has been suggested. The implementation of the suggested methodologies on graphics processing units (GPUs) with multiple cores offers promise for further boosting performance and obtaining faster processing rates; this might be a viable path for additional study in the future.

## 5.2 Future Works

In The future, the following direction can be considered:

1- The encryption key can be generated using-different methods.

2- Executing the proposed message authentication algorithm over GPU to see how the proposed methods can be accelerated.

## Reference

1. Noura, Hassan N., et al. "DKEMA: GPU-based and dynamic key-dependent efficient message authentication algorithm." *The Journal of Supercomputing* 78.12 (2022): 14034-14071.
2. Lal, N. A., Prasad, S., & Farik, M. A review of authentication methods." *International journal of scientific & technology research*, (2016), 246-249.
3. Roberts, Amanda, Steve Sharman, and Henrietta Bowden-Jones. "Clinical services for problematic internet usage." *Current Opinion in Behavioral Sciences* 46 (2022): 101180.
4. Fanfakh, Ahmed, Hassan Noura, and Raphaël Couturier. "ORSCA-GPU: one round stream cipher algorithm for GPU implementation." *The Journal of Supercomputing* 78.9 (2022): 11744-11767.
5. Saarinen, M. J., & Aumasson, J. P. The BLAKE2 cryptographic hash and message authentication code (MAC) (2015): (No. rfc7693).
6. Wang, Ziheng, et al. "Parallel SHA-256 on SW26010 many-core processor for hashing of multiple messages." *The Journal of Supercomputing* 79.2 (2023): 2332-2355.
7. Abdellatif, Karim M., Roselyne Chotin-Avot, and Habib Mehrez. "Efficient parallel-pipelined GHASH for message authentication." *2012 International Conference on Reconfigurable Computing and FPGAs*. IEEE, 2012.
8. Pirzada, Syed Jahanzeb Hussain, et al. "The parallel CMAC authentication Algorithm." *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2019.

9. S. J. H. Pirzada, A. Murtaza, and J. Liu,“Implementation of CMAC Authentication Algorithm on FPGA for SatelliteCommunication”, In Proc. of IEEE 3rd Information Technology, Networking,Electronic and Automation Control Conference (ITNEC), pp. 1-5, 2019.
10. Pirzada, Syed Jahanzeb Hussain, Abid Murtaza, and Jianwei Liu. "Implementation of CMAC Authentication Algorithm on FPGA for Satellite Communication." *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2019.
11. Ravilla, Dilli, and Chandra Shekar Reddy Putta. "Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs." *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*. IEEE, 2015.
12. Fanfakh, Ahmed, Hassan Noura, and Raphaël Couturier. "Simultaneous encryption and authentication of messages over GPUs." *Multimedia Tools and Applications* (2023): 1-33.
13. Shigeri, Maki, and Hiroyasu Kubo. "Parallel Verification of Serial MAC and AE Modes." *Selected Areas in Cryptography: 28th International Conference, Virtual Event, September 29–October 1, 2021, Revised Selected Papers*. Vol. 13203. Springer Nature, 2022.
14. Krovetz, Ted. *UMAC: Message authentication code using universal hashing*. No. rfc4418. 2006.
15. Pammu, Ali Akbar, et al. "A high throughput and secure authentication-encryption AES-CCM algorithm on asynchronous multicore processor." *IEEE Transactions on Information Forensics and Security* 14.4 (2018): 1023-1036.

16. Bokhari, Mohammad Ubaidullah, and Qahtan Makki Shallal. "A review on symmetric key encryption techniques in cryptography." *International journal of computer applications* 147.10 (2016).
17. Fox, Geoffrey C., Roy D. Williams, and Guiseppe C. Messina. *Parallel computing works.* Elsevier, 2014.
18. Diffie, Whitfield, and Martin E. Hellman. "New directions in cryptography." *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022. 365-390.
19. KILIÇ, Muhammed Burak. "Encryption methods and comparison of popular chat applications." *Advances in Artificial Intelligence Research* 1.2 (2021): 52-59.
20. Alenezi, Mohammed N., Haneen Alabdulrazzaq, and Nada Q. Mohammad. "Symmetric encryption algorithms: Review and evaluation study." *International Journal of Communication Networks and Information Security* 12.2 (2020): 256-272.
21. Zhang, Qixin. "An overview and analysis of hybrid encryption: the combination of symmetric encryption and asymmetric encryption." *2021 2nd international conference on computing and data science (CDS)*. IEEE, 2021.
22. Bhanot, Rajdeep, and Rahul Hans. "A review and comparative analysis of various encryption algorithms." *International Journal of Security and Its Applications* 9.4 (2015): 289-306.
23. Abdullah, Ako Muhamad. "Advanced encryption standard (AES) algorithm to encrypt and decrypt data." *Cryptography and Network Security* 16.1 (2017): 11.
24. Çavuşoğlu, Ünal, et al. "The design and implementation of hybrid RSA algorithm using a novel chaos based RNG." *Chaos, Solitons & Fractals* 104 (2017): 655-667.

25. Adhie, Roy Pramono, et al. "Implementation cryptography data encryption standard (DES) and triple data encryption standard (3DES) method in communication system based near field communication (NFC)." *Journal of Physics: Conference Series*. Vol. 954. No. 1. IOP Publishing, 2018.
26. Buchanan, William J., Shancang Li, and Rameez Asif. "Lightweight cryptography methods." *Journal of Cyber Security Technology* 1.3-4 (2017): 187-201.
27. Rana, Muhammad, Quazi Mamun, and Rafiqul Islam. "Lightweight cryptography in IoT networks: A survey." *Future Generation Computer Systems* 129 (2022): 77-89.
28. Lustro, Roman Alex F., Ariel M. Sison, and Ruji P. Medina. "Performance analysis of enhanced SPECK algorithm." *Proceedings of the 4th International Conference on Industrial and Business Engineering*. 2018.
29. Alassaf, Norah, et al. "Enhancing speed of SIMON: A light-weight-cryptographic algorithm for IoT applications." *Multimedia Tools and Applications* 78 (2019): 32633-32657.
30. Santoli, Thomas, and Christian Schaffner. "Using Simon's algorithm to attack symmetric-key cryptographic primitives." *arXiv preprint arXiv:1603.07856* (2016).
31. El Hennawy, Hadia MS, Alaa EA Omar, and Salah MA Kholaif. "LEA: link encryption algorithm proposed stream cipher algorithm." *Ain Shams Engineering Journal* 6.1 (2015): 57-65.
32. Song, Ling, Zhangjie Huang, and Qianqian Yang. "Automatic differential analysis of ARX block ciphers with application to SPECK and LEA." *Australasian Conference on Information Security and Privacy*. Cham: Springer International Publishing, 2016.

33. Sumartono, Isnar, Andysah Putera Utama Siahaan, and Nova Mayasari. "An overview of the RC4 algorithm." *IOSR J. Comput. Eng* 18.6 (2016): 67-73.
34. Barkadehi, Mohammadreza Hazhirpasand, et al. "Authentication systems: A literature review and classification." *Telematics and Informatics* 35.5 (2018): 1491-1511.
35. Chenchev, Ivaylo, Adelina Aleksieva-Petrova, and Milen Petrov. "Authentication Mechanisms and Classification: A Literature Survey." *Intelligent Computing: Proceedings of the 2021 Computing Conference, Volume 3*. Springer International Publishing, 2021.
36. Kashyap, Ramgopal. "Biometric authentication techniques and E-learning." *Biometric Authentication in Online Learning Environments*. IGI Global, 2019. 236-265.
37. Cilia, Darren, and Frankie Inguanez. "Multi-model authentication using keystroke dynamics for smartphones." *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2018.
38. Lozupone, Vincent. "Analyze encryption and public key infrastructure (PKI)." *International Journal of Information Management* 38.1 (2018): 42-44.
39. Niewolski, Wojciech, et al. "Token-based authentication framework for 5G MEC mobile networks." *Electronics* 10.14 (2021): 1724.
40. Velásquez, Ignacio, Angélica Caro, and Alfonso Rodríguez. "Authentication schemes and methods: A systematic literature review." *Information and Software Technology* 94 (2018): 30-37.

41. Chi, Lianhua, and Xingquan Zhu. "Hashing techniques: A survey and taxonomy." *ACM Computing Surveys (Csur)* 50.1 (2017): 1-36.
42. Sobti, Rajeev, and Ganesan Geetha. "Cryptographic hash functions: a review." *International Journal of Computer Science Issues (IJCSI)* 9.2 (2012): 461.
43. Estébanez, César, et al. "Performance of the most common non-cryptographic hash functions." *Software: Practice and Experience* 44.6 (2014): 681-698.
44. Khakim, L., M. Mukhlisin, and A. Suharjono. "Security system design for cloud computing by using the combination of AES256 and MD5 algorithm." *IOP Conference Series: Materials Science and Engineering*. Vol. 732. No. 1. IOP Publishing, 2020.
45. Khan, Burhan Ul Islam, et al. "Evolution and analysis of secured hash algorithm (SHA) family." *Malaysian Journal of Computer Science* 35.3 (2022): 179-200.
46. Chaves, Ricardo, et al. "Secure hashing: Sha-1, sha-2, and sha-3." *Circuits and systems for security and privacy* (2016): 105-132.
47. Hockney, Roger W., and Chris R. Jesshope. *Parallel Computers 2: architecture, programming and algorithms*. CRC Press, 2019.
48. Hepola, Kari, Joonas Multanen, and Pekka Jääskeläinen. "Dual-IS: Instruction Set Modality for Efficient Instruction Level Parallelism." *International Conference on Architecture of Computing Systems*. Cham: Springer International Publishing, 2022.
49. Lin, Zhen, Michael Mantor, and Huiyang Zhou. "GPU performance vs. thread-level parallelism: Scalability analysis and a novel way to improve TLP." *ACM Transactions on Architecture and Code Optimization (TACO)* 15.1 (2018): 1-21.

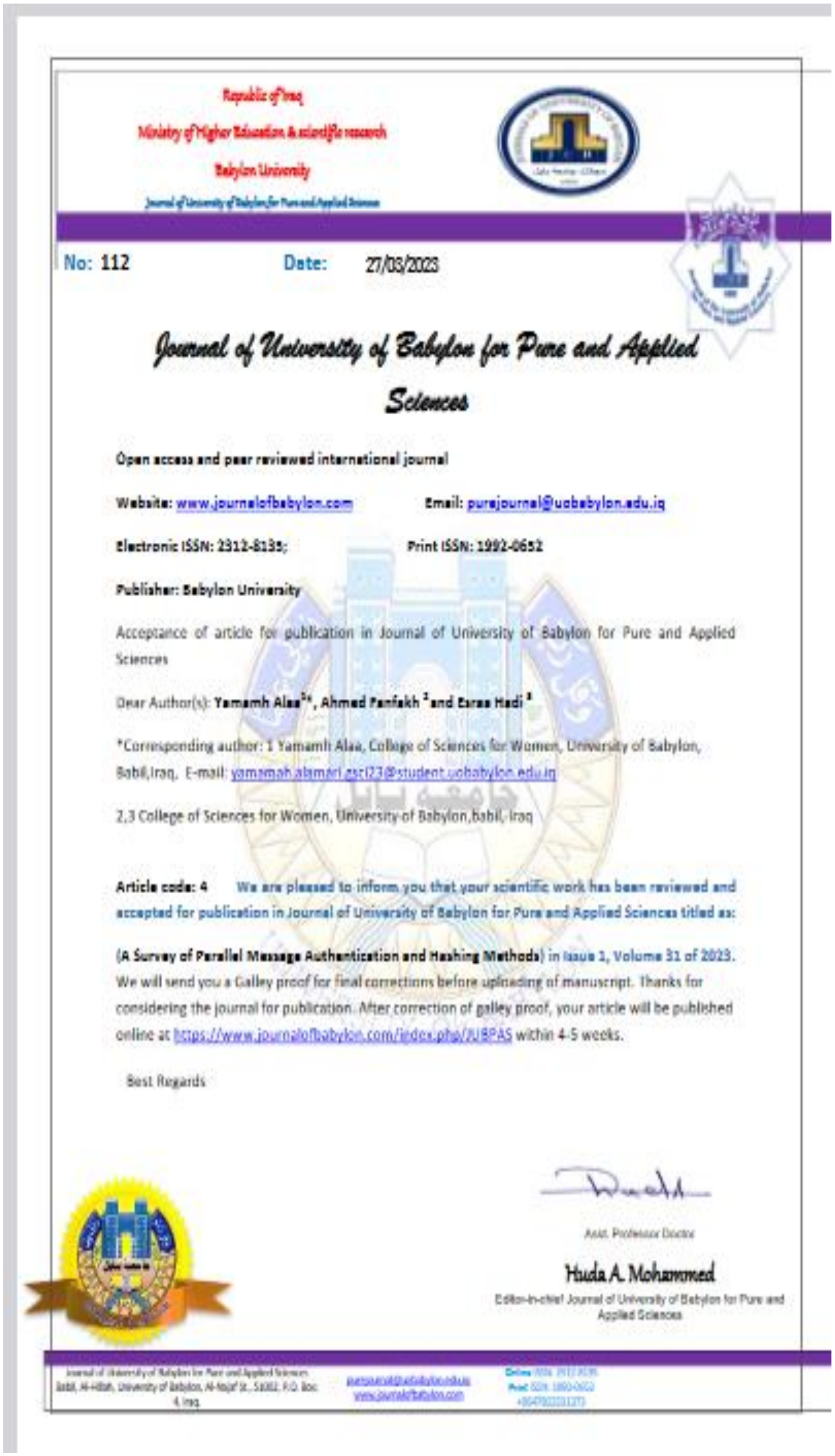
50. Azarian, Ali. *Task-Level Pipelining in Configurable Multicore Architectures*. Diss. Universidade do Porto (Portugal), 2016.
51. Weisz, Gabriel, et al. "A study of pointer-chasing performance on shared-memory processor-FPGA systems." *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2016.
52. Martin, Christian. "Multicore processors: challenges, opportunities, emerging trends." *Proc. Embedded World Conference*. Vol. 2014. 2014.
53. Zhuravlev, Sergey, et al. "Survey of scheduling techniques for addressing shared resources in multicore processors." *ACM Computing Surveys (CSUR)* 45.1 (2012): 1-28.
54. YarKhan, Asim. "Dynamic task execution on shared and distributed memory architectures." (2012).
55. Yu, Jia, Jinxuan Wu, and Mohamed Sarwat. "Geospark: A cluster computing framework for processing large-scale spatial data." *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*. 2015.
56. Ivanov, Andrey M., and Nikolay Igorevich Khokhlov. "Parallel implementation of the grid-characteristic method in the case of explicit contact boundaries." *Computer research and modeling* 10.5 (2018): 667-678.
57. Barker, Brandon. "Message passing interface (mpi)." *Workshop: high performance computing on stampede*. Vol. 262. Houston, TX, USA: Cornell University Publisher, 2015.
58. Byrne, Simon, Lucas C. Wilcox, and Valentin Churavy. "MPI. jl: Julia bindings for the Message Passing Interface." *Proceedings of the JuliaCon Conferences*. Vol. 1. No. 1. 2021.

59. Ferguson, Andrew L. "BayesWHAM: A Bayesian approach for free energy estimation, reweighting, and uncertainty quantification in the weighted histogram analysis method." *Journal of computational chemistry* 38.18 (2017): 1583-1605.
60. Xie, Eric Yong, et al. "On the cryptanalysis of Fridrich's chaotic image encryption scheme." *Signal processing* 132 (2017): 150-154.
61. Abutaha, Mohammed, Islam Amar, and Salman AlQahtani. "Parallel and practical approach of efficient image chaotic encryption based on message passing interface (MPI)." *Entropy* 24.4 (2022): 566.
62. Noura, Hassan, et al. "One round cipher algorithm for multimedia IoT devices." *Multimedia tools and applications* 77 (2018): 18383-18413.
63. Dewangan, Chandra Prakash, et al. "Study of avalanche effect in AES using binary codes." *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*. IEEE, 2012.
64. Doty-Humphrey C (2014) Practrand. http://pracrand.sourceforge.net
65. Yalta, A. Talha, and Sven Schreiber. "Random number generation in gretl." *Journal of Statistical Software* 50 (2012): 1-13.
66. Brown, Robert G., Dirk Eddelbuettel, and David Bauer. "Dieharder." *Duke University Physics Department Durham, NC* (2018): 27708-0305.
67. Hernandez-Castro, Julio, and David F. Barrero. "Evolutionary generation and degeneration of randomness to assess the indepedence of the Ent test battery." *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017.

68. Hegadi, Rajendra, and Abhijit Prakash Patil. "A Statistical Analysis on In-Built Pseudo Random Number Generators Using NIST Test Suite." *2020 5th International Conference on Computing, Communication and Security (ICCCS)*. IEEE, 2020.
69. Asassfeh, Mahmoud Rahallah, Mohammad Qatawneh, and Feras Mohamed AL-Azzeh. "Performance evaluation of blowfish algorithm on supercomputer iman1." *International Journal of Computer Networks & Communications (IJCNC)* 10.2 (2018).
70. Tang, Shanjiang, Bu-Sung Lee, and Bingsheng He. "Speedup for multi-level parallel computing." *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 2012.

## Publications

1. Alamari, Yamamh Alaa, Ahmed Fanfakh, and Esraa Hadi. "Parallel Message Authentication Algorithm Implemented Over Multicore CPU." *International Journal of Intelligent Engineering & Systems* 16.4 (2023).
2. Alaa, Yamamh, Ahmed Fanfakh, and Esraa Hadi. "A Survey of Parallel Message Authentication and Hashing Methods." *Journal of University of Babylon for Pure and Applied Sciences* (2023): 100-110.



Republic of Iraq

Ministry of Higher Education & scientific research

Babylon University

Journal of University of Babylon for Pure and Applied Sciences

No: 112 Date: 27/03/2023

# *Journal of University of Babylon for Pure and Applied Sciences*

**Open access and peer reviewed international journal**

**Website:** www.journalofbabylon.com **Email:** purejournal@uobabylon.edu.iq

**Electronic ISSN: 2312-8135; Print ISSN: 1992-0652**

**Publisher: Babylon University**

Acceptance of article for publication in Journal of University of Babylon for Pure and Applied Sciences

Dear Author(s): **Yamamh Alaa[1*], Ahmed Fanfakh [2] and Esraa Hadi [3]**

*Corresponding author: 1 Yamamh Alaa, College of Sciences for Women, University of Babylon, Babil,Iraq, E-mail: yamamah.alamari.gsci23@student.uobabylon.edu.iq

2,3 College of Sciences for Women, University of Babylon,babil,-Iraq

**Article code: 4** **We are pleased to inform you that your scientific work has been reviewed and accepted for publication in Journal of University of Babylon for Pure and Applied Sciences titled as:**

**(A Survey of Parallel Message Authentication and Hashing Methods)** in Issue 1, Volume 31 of 2023. We will send you a Galley proof for final corrections before uploading of manuscript. Thanks for considering the journal for publication. After correction of galley proof, your article will be published online at https://www.journalofbabylon.com/index.php/JUBPAS within 4-5 weeks.

Best Regards

[illegible]

Asst. Professor Doctor

Huda A. Mohammed

Editor-in-chief Journal of University of Babylon for Pure and Applied Sciences

Journal of University of Babylon for Pure and Applied Sciences Babil, Al-Hillah, University of Babylon, Al-Najaf St., 51002, P.O. Box: 4, Iraq.

purejournal@uobabylon.edu.iq
www.journalofbabylon.com

Online ISSN: 2312-8135
Print ISSN: 1992-0652
+9647[illegible]

## الملخص

يوجد حوالي 4.95 مليار مستخدم للإنترنت في جميع أنحاء العالم في الوقت الحاضر ، مما يجعل هذا السوق سوقًا كبيرًا مع طلب متزايد باطراد على الخدمات عبر الإنترنت. يتضمن ذلك أشياء مثل التواصل الاجتماعي عبر الإنترنت ومشاركة المعلومات وكذلك الشراء عبر الإنترنت. هذا يسلط الضوء على المتطلبات الحاسمة لمزيد من الخصوصية والسرية. برز الاحتيال عبر الإنترنت كواحد من أهم العوائق أمام الاستخدام الواسع النطاق لتطبيقات الأعمال. وقد أدى ذلك إلى ظهور مخاوف بشأن المصادقة والترخيص وتحديد الهوية باعتبارها تحديات مهمة في مجتمعنا الحديث والمفتوح. إجراء تحديد الهوية هو عملية التعرف على شخص أو كمبيوتر أو تطبيق برمجي. تستخدم أنظمة الأمان المصادقة والترخيص معًا لتحديد من يُسمح له بالوصول إلى البيانات والأنظمة عبر الشبكة. أداة المستخدم هي المسؤولة عن المصادقة ، بينما يكون نظام الأمان مسؤولاً عن التفويض. ومع ذلك ، فقد تم اقتراح التوازي كوسيلة لزيادة فعالية إجراءات المصادقة من بين البدائل المحتملة الأخرى. في هذه الأطروحة ، شرعنا في إنشاء خوارزمية جديدة تمامًا لمصادقة الرسائل بالتوازي عبر عدة مراكز للكمبيوتر. يتم استخدام ملفي PNGRومربعين بديلين في إجراء مصادقة الرسالة المقترح لتشفير الرسالة العادية والتحقق منها. يتطلب التشفير جولة واحدة فقط لتشفير أو فك تشفير كتلة من البيانات ، مما يجعلها طريقة فعالة للغاية. عند تنفيذها على وحدة معالجة مركزية متعددة النواة ، تكون طريقة المصادقة المقترحة أسرع 3.27 مرة من طريقة المصادقة المتوازية القائمة على النقطة. عند مقارنة الخوارزمية المقترحة للتنفيذ المتوازي مع نظيرتها التسلسلية ، يتم تحقيق متوسط تسريع يبلغ 2.99. تبين أن الطريقة المقترحة عشوائية بدرجة كافية لاجتياز الاختبار الأكثر صرامة ، كما تبين أيضًا أن قيم MAC الناتجة اجتازت أصعب اختبار للأمان

وزارة التعليم العالي و البحث العلمي

جامعة بابل كلية العلوم للبنات

قسم علوم الحاسوب

# تنفيذ خوارزمية رمز مصادقة الرسالة في بيئة متعددة النواة

رسالة مقدمة الى مجلس كلية العلوم للبنات في جامعة بابل وهي جزء من متطلبات الحصول على درجة الماجستير في علوم الحاسبات

مقدمة من قبل

**يمامه علاء عبد الكاظم**

باشراف

الاستاذ المساعد

**احمد بدري مسلم**

الاستاذ المساعد

**اسراء هادي عبيد**

**1445 هـ** **2023 م**