وزارة التعليم العالي و البحث العلمي

جامعة بابل

كلية علوم البنات

قسم علوم الحاسوب

Enhancing Data Security: Implementing the Present Lightweight Algorithm for Encrypting

بحث تقدمت به الطالبة:

نور حميد محمد

كجزء من متطلبات نيل درجة البكالوريوس

الاشراف:

م.م. زهراء عبد محمد

2024 - 2023

بسم الله الرحمن الرحيم

الاهـــــــــــــــداء

الى خالق اللوح و القلم و باري الذر و النسم و خالق كل شيء من العدم

الى من بلغ الرسالة و ادى الامانة و نص الامة ..الى نبي الرحمة

الى السادات الاطهار و عروته الوثقى.. اهل بيت النبوة

الى مراد قلبي و الاقرب لي من نفسي المغيب عن الابصار.. صاحب العصر و الزمان (عج)

الى من علمني ان الدنيا كفاح و سلاحها العلم و المعرفة الى الذي لم يبخل علية باي شيء الى من سعى لاجل راحتي و نجاحي ...ابي العزيز

الى ذات القلب النقي ..

الى من اوصاني بها الرحمن برا و احسانا الى من سعت و عانت من اجلي ..امي الحبيبة

الى من اشاركهم لحظاتي و يفرحون لنجاحي اخوتي و اساتذتي الكرام ..

اهديكم هذا الجهد المتواضع

الشكر والتقدير

نحمد الله الذي وفقنا في اتمام هذا البحث العلمي و الذي الهمنا الصحة و العافية و العزيمة .

فالحمد لله حمدا كثيرا

نتقدم بجزيل الشكر و التقدير الى

الست زهراء عبد محمد

على كل ما قدمته لنا من معلومات و توجيهات قيمه ساهمت في اثراء موضوع دراستنا في جوانبها المختلفة. .

كما نتقدم بجزيل الشكر الى اعضاء لجنة المناقشة

LIST OF CONTENTS

List of Figures

ABSTRACT

In today's digital era, ensuring the confidentiality and integrity of data has become paramount. With the exponential growth of data generation and transmission, the need for efficient yet secure encryption algorithms has intensified. This study investigates the efficacy of the Present lightweight algorithm in encrypting text files,

The Present algorithm is renowned for its simplicity, efficiency, and robustness against various cryptographic attacks. Its lightweight nature makes it suitable for resource-constrained environments without compromising security. In this research, we analyze the performance of the Present algorithm when applied to encrypting a text file of significant size

Through extensive experimentation and analysis, we evaluate the encryption speed, computational overhead, and security strength of the Present algorithm in handling a 1MB text file. Furthermore, we assess its resistance against common cryptographic attacks, such as brute force and differential cryptanalysis

The results demonstrate that the Present algorithm efficiently encrypts text files while maintaining a high level of security. It exhibits commendable encryption speed and minimal computational overhead, making it suitable for real-time applications and systems with limited processing capabilities. Moreover, the algorithm showcases resilience against various cryptographic attacks, affirming its suitability for safeguarding sensitive data in diverse domains, including communication, finance, and healthcare

This research contributes to the advancement of data security by providing empirical evidence of the effectiveness of the Present lightweight algorithm in encrypting text files. It underscores the algorithm's viability as a practical solution for securing data in contemporary digital environments, where the protection of information assets is of utmost importance.

CHAPTER ONE

1.1 Introduction

Lightweight cryptography has gained significant attention due to the proliferation of Internet of Things (IoT) devices, wearable technologies, and other resource-constrained environments where traditional cryptographic algorithms may be impractical or too resource-intensive. Lightweight algorithms are specifically designed to offer robust security with minimal computational and memory requirements, making them ideal for embedded systems with limited processing capabilities

The primary objectives of lightweight cryptography are to ensure data confidentiality, integrity, and authentication while maintaining high performance on devices with restricted resources. These algorithms often employ streamlined designs, reduced key sizes, and optimized operations to achieve efficient cryptographic operations without compromising security

Several lightweight cryptographic algorithms have been proposed and standardized to address the unique challenges posed by resource-constrained environments. These algorithms encompass various cryptographic primitives, including block ciphers, hash functions, and authentication protocols, tailored to meet the specific needs of lightweight applications.

1.2 Problem Statement

In today's digital landscape, the proliferation of data across various platforms and networks has heightened concerns regarding data security. The need to safeguard sensitive information from unauthorized access and malicious attacks is more critical than ever. To address these challenges, encryption techniques play a pivotal role in ensuring the confidentiality and integrity of data during transmission and storage

However, traditional encryption algorithms often face limitations when applied to resource-constrained environments or when dealing with volumes of data. These limitations can include high computational overhead, increased processing time, and a lack of scalability. Consequently, there arises a need for lightweight encryption algorithms that can efficiently secure data without compromising performance

The Present lightweight algorithm has emerged as a promising solution to these challenges. With its simplicity, efficiency, and strong cryptographic properties.

1.3 The Goal of "Lightweight Algorithm"

The Present algorithm aims to achieve many goals in the encryption and data protection process, including

Encryption speed The Present algorithm is designed to be efficient in terms of speed in encryption and decryption operations. This means that it is capable of processing large amounts of data very efficiently

Power of protection The Present algorithm provides a high level of security and protection for encrypted data. Advanced technologies are used to prevent any attempt to hack encrypted data

Data PreservationThe Present algorithm encrypts data in a structured and .secure manner, which helps save data from unauthorized access

Ease of use The Present algorithm is easy to use, making it suitable for use in a variety of applications and systems

In short, it can be said that the main goal of using the Present algorithm in encryption is to provide high security, speed in encryption and decryption processes, and to keep data securely and easily in use

CHAPTER TWO

2.1 Introduction

Java is a widely-used, high-level programming language known for its platform independence, robustness, and versatility. Developed by Sun Microsystems (now owned by Oracle Corporation) in the mid-1990s, Java was designed with a focus on simplicity, object-oriented programming, and portability, making it a popular choice for developing a wide range of applications, from enterprise-level systems to mobile and web applications

One of Java's key features is its "Write Once, Run Anywhere" principle, facilitated by the Java Virtual Machine (JVM). Java source code is compiled into bytecode, which is platform-independent and can be executed on any device that has a JVM, eliminating the need for recompilation and allowing Java applications to run on various operating systems, including Windows, macOS, Linux, and more

Java offers a rich standard library, providing developers with a vast collection of pre-built classes and methods to simplify common programming tasks and accelerate development. Additionally, its strong type system, automatic memory management through garbage collection, and exception handling mechanisms contribute to creating robust and reliable applications

Over the years, Java has evolved and adapted to the changing technological landscape, introducing new features and enhancements to support modern development practices, such as functional programming with lambda expressions, modular programming with the Java Platform Module System (JPMS), and improved performance optimizations

2.2 Lightweight Overview

The applications and services that need secure communication, they also need speed in completing commands. In the end, these applications operate on the principle of real time. This is not achieved by traditional encryption algorithms, which require high costs of memory, processors and energy. So, it is lightweight algorithms that will solve this problem [5]. Therefore, NIST, “the National Institute of Standards and Technology”, and (ISO / IEC), “the International Organization for Standardization / International Electrotechnical Commission”, provided a set of conditions for the encryption algorithm to be considered suitable for real-time applications. Limited memory and are battery powered devices are the properties of these devices. Often these devices have limited memory and not large, in addition to their limited capacity, as they work on batteries. Therefore, well-known conventional methods such as AES may not be useful for these devices. AES method, for example, is a good method for security, but it cannot work on these devices because it requires a lot of memory. So, the solution was that NIST had arrived using algorithms called lightweight algorithms that had a high degree of security [3, 6]. In the context of talking about this topic, NIST launched an initiative some time ago to standardize and evaluate lightweight algorithms for coding. Whereas, NIST is trying to address existing standards that are no longer acceptable. NIST has invited researchers to submit their algorithms for consideration according to criteria, and the submission period has expired. The accepted algorithms went through several qualifiers until the moment of writing the paper reached 27 methods. NIST has announced that the date to announce the algorithms candidates for the final qualifiers will be announced at the end of February 2021. It also asked those interested in the topic to enter its site and add their comments to each algorithm, and it will take into account all comments

2.3 ENCRYPTION

Encryption is a method used to secure information by converting it into an unreadable format, known as cipher text, using an encryption algorithm and an encryption key. This process ensures that only authorized parties can access and understand the original information

There are two main types of encryption: symmetric and asymmetric

1. Symmetric Encryption: In symmetric encryption, the same key is used for both encryption and decryption. This means that the sender and the receiver must share the same secret key. Popular symmetric encryption algorithms include AES (Advanced Encryption Standard), .(DES (Data Encryption Standard), and 3DES (Triple DES

2. Asymmetric Encryption: Asymmetric encryption uses a pair of keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption. This allows for secure communication without sharing a secret key beforehand. RSA (Rivest -Shamir- Adleman) and ECC (Elliptic Curve Cryptography) are common asymmetric encryption algorithms.

2.4 Decryption

is the process of converting encrypted data or cipher text back into its original, readable form using the appropriate decryption key. It's the reverse operation of encryption and is essential for retrieving and .understanding the original information after it has been encrypted

There are two primary types of decryption methods, depending on the encryption technique used

1.Symmetric Decryption: In symmetric decryption, the same secret key used for encryption is also used for decryption. Both the sender and receiver need to have access to this secret key to encrypt and decrypt messages. The symmetric decryption algorithm performs the inverse operation of the encryption algorithm, transforming cipher text back into .plaintext

2.Asymmetric Decryption: Asymmetric decryption involves two different keys: a public key and a private key. The cipher text, encrypted with the public key, can only be decrypted using the corresponding private key. This method allows for secure communication without sharing a secret key beforehand. The most common asymmetric .decryption algorithms include RSA and ECC

:Steps to Decryption

1. Select the Correct Decryption Key: Ensure you have the correct .decryption key that matches the encryption key used to encrypt the data

2. Apply the Decryption Algorithm: Use the appropriate decryption algorithm along with the decryption key to transform the cipher text back .into plaintext

3. Retrieve the Original Data: Once the decryption process is completed, the original data or plaintext will be obtained, making it .readable and understandable

is the process of converting encrypted data or cipher text back into its original, readable form using the appropriate decryption key. It's the reverse operation of encryption and is essential for retrieving and understanding the original information after it has been encrypted

There are two primary types of decryption methods, depending on the :encryption technique used

1.Symmetric Decryption: In symmetric decryption, the same secret key used for encryption is also used for decryption. Both the sender and receiver need to have access to this secret key to encrypt and decrypt messages. The symmetric decryption algorithm performs the inverse operation of the encryption algorithm, transforming cipher text back into .plaintext

2.Asymmetric Decryption: Asymmetric decryption involves two different keys: a public key and a private key. The cipher text, encrypted with the public key, can only be decrypted using the corresponding private key. This method allows for secure communication without sharing a secret key beforehand. The most common asymmetric .decryption algorithms include RSA and ECC

Steps to Decryption:

1. Select the Correct Decryption Key: Ensure you have the correct .decryption key that matches the encryption key used to encrypt the data

2. Apply the Decryption Algorithm: Use the appropriate decryption algorithm along with the decryption key to transform the cipher text back .into plaintext

3. Retrieve the Original Data: Once the decryption process is completed, the original data or plaintext will be obtained, making it readable and understandable.

It's crucial to keep decryption keys secure and confidential, just like encryption keys. Unauthorized access to the decryption key can compromise the security of the encrypted data and expose sensitive .information

Decryption plays a vital role in cyber security, ensuring that authorized parties can access and understand encrypted data while keeping it .protected from unauthorized access and potential threats

2.5 Lightweight Algorithms Example:

symmetric type of cipher. The entry of information into encryption methods of this type is in the form of blocks (for example, 64 or 128 bits). Of the methods defined in this type that are lightweight are: Bibliography, Glossary, Katan , Clefia and Present. The other group of symmetric encryption methods is stream cipher. This type of encryption forms the information entering the algorithm in a flow rather than in blocks. This means that the data is entered bit by bit. Examples of this type that are lightweight algorithms are: LEA,PRESENT,AES

2.5.1 LEA

The LEA algorithm, known as a light encryption algorithm, is a block cipher algorithm. It started in 128-bit format and was used to provide confidentiality and protection to cloud computing that is defined as high-speed environments. It has also been used in mobile devices and the Internet of things. The data blocks in this algorithm can be in three shapes which are 128, 192 and finally 256 bits. It has been tested and found to be faster than AES algorithm, from 1.5 to 2. This algorithm has been certified and considered within the standards of ISO / IEC 29192- 2: 2019 which provides for information security and lightweight coding [8]. 4.2.

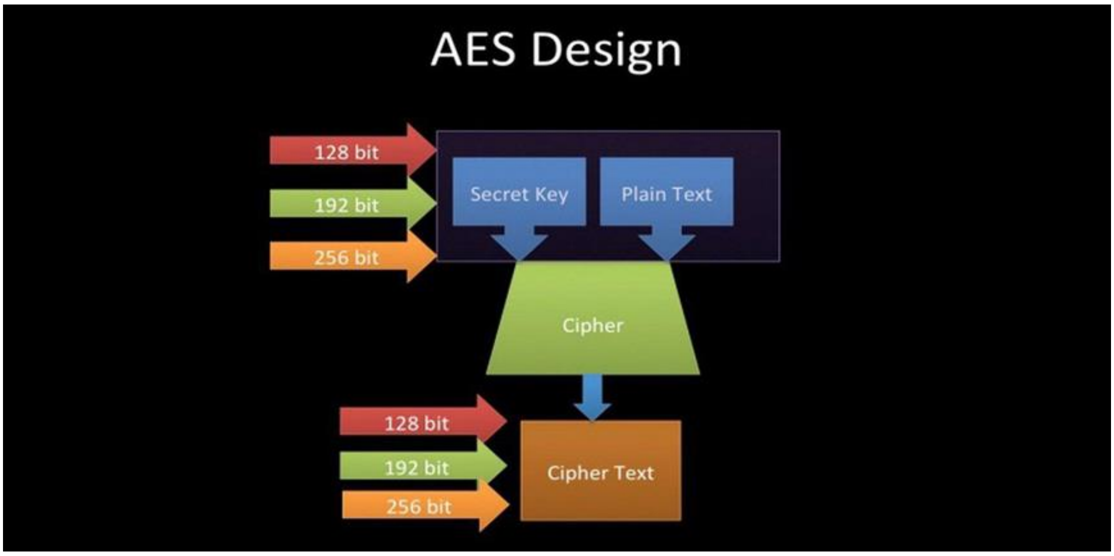
2.5.2 AES

AES (Advanced Encryption Standard) algorithm is an advanced encryption algorithm widely used in protection and security. AES was developed as an alternative to the older DES (Data Encryption Standard) algorithm due to its high robustness and resistance to attacks.

AES uses three different key sizes: 128-bit, 192-bit, and 256-bit. Starting from plaintext, the encryption process in AES takes place through several rounds of Substitute Bytes, Shift Rows, Mix Columns, and Add Round Key.

During the encryption rounds, the secret key is combined in a complex manner with the clear text data, and the encryption processes are repeated for several rounds (10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key) to encrypt the data effectively and securely.

Thanks to its strong encryption and resistance to attacks, the AES algorithm is used for a wide range of



2.5.3 PRESENT

It is one of the lightweight encryption methods that was developed in France by the Orange Laboratory as well as the German University Bochum in addition to the Technical University of Denmark in 2007. It is an algorithm characterized by small size as it is 2.5 times smaller than AES [9]. The size of the input data is 64 bits, and it has been developed to be either 80 or 128 bits. Also, in this method, the single S-Box is only 4-bit by 4-bit size. It is also known that the goal of lightweight algorithms is for safety with less power consumption and less memory, and this is what the PRESENT method has been incorporated into it by IOS / IES [5]. Figure 1 showing the Structure of PRESENT algorithm

2.6 Methodology

PLA Implementation: Develop a Java implementation of the PLA based on its specifications. Implement encryption and decryption functions .capable of processing text files efficiently

Scalability Evaluation: Create a testing framework to assess PLA's scalability by encrypting and decrypting text files of increasing sizes. Measure the algorithm's performance in terms of memory usage and .processing time

Performance Analysis: Benchmark PLA's encryption and decryption speeds using different file sizes. Analyze the results to identify potential .bottlenecks and optimize the algorithm for improved performance

Security Assessment: Conduct a comprehensive security analysis to evaluate PLA's resistance against brute force, differential cryptanalysis, and other potential attacks. Validate the algorithm's cryptographic .strength when applied to text files

Integration and Testing: Integrate the PLA implementation into a user-friendly application with intuitive interfaces for file encryption and decryption. Perform extensive testing, including unit tests, integration tests, and user acceptance tests, to ensure the application's functionality and reliability.

CHAPTER THREE

3.1 Introduction

Present offers a viable option for encrypting text data in scenarios where computational resources are limited or where fast encryption and decryption are required. However, despite its potential benefits, there remain several key issues and challenges that need to be addressed when .employing the Present algorithm for text encryption

One significant challenge is the scalability of the Present algorithm when encrypting text files. While Present is known for its efficiency in small-scale applications, its performance may degrade when tasked with encrypting text files of considerable size, such as those exceeding 1MB. This scalability issue can lead to increased encryption times and computational overhead, potentially impacting the overall system .performance and user experience

Moreover, there may be concerns regarding the security strength of the Present algorithm when applied to text encryption. Although Present is designed to resist various cryptographic attacks, including differential and linear cryptanalysis, its effectiveness against sophisticated attacks on text files warrants further investigation. Ensuring the robustness and resilience of Present in real-world scenarios with extensive text data is .essential for its widespread adoption and deployment

Therefore, the primary objective of this study is to address the scalability and security challenges associated with using the Present lightweight algorithm for encrypting text data, particularly focusing on files. By conducting comprehensive analyses and experiments, this research aims to evaluate the performance, efficiency, and security of Present in handling text files, thereby providing valuable insights into its practical applicability and effectiveness as a data encryption solution.

3.2 Design Proposed Approach

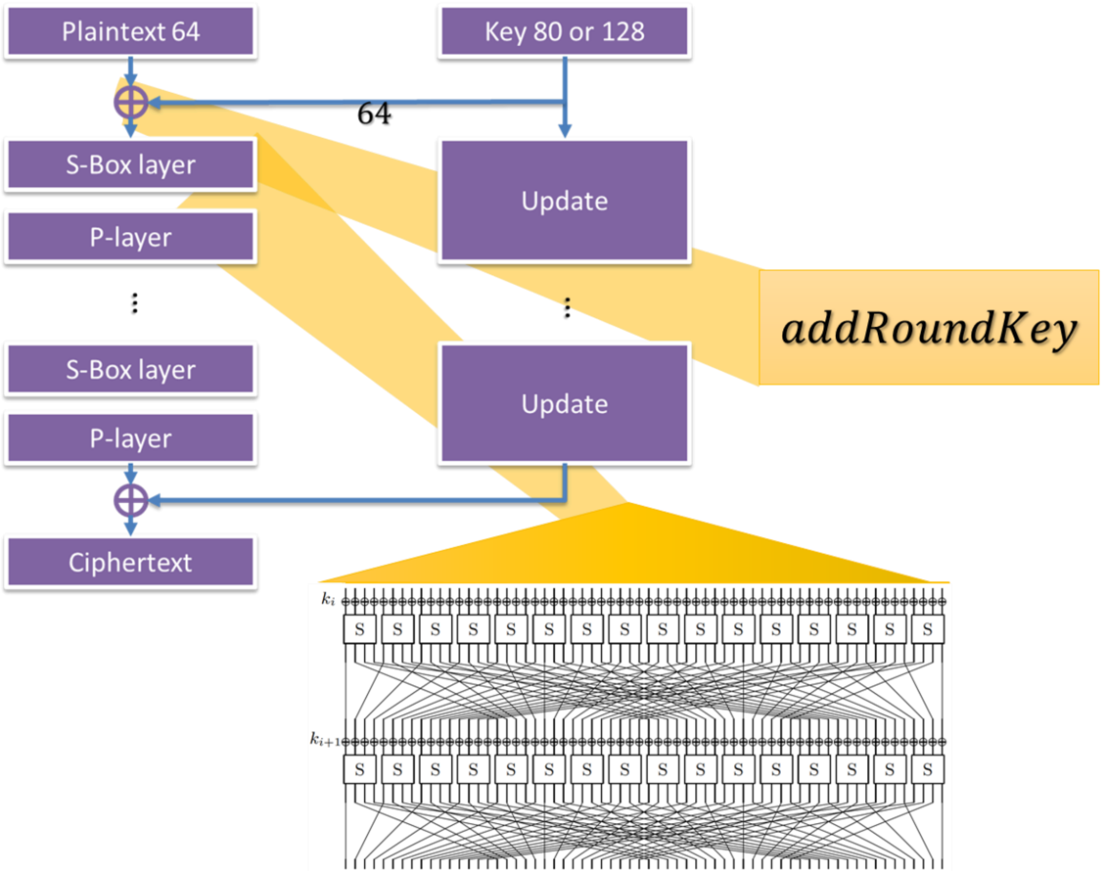
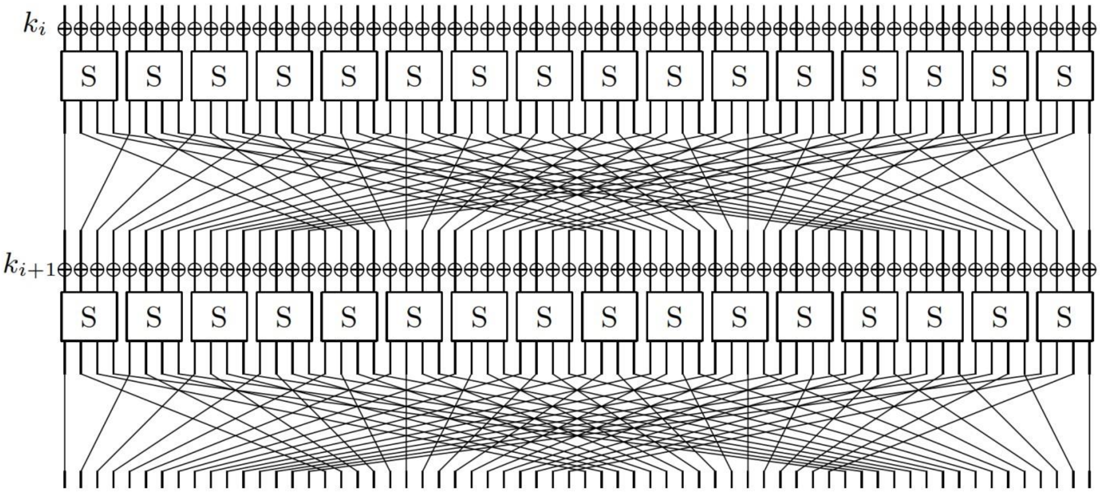


Figure 1: Structure of PRESENT

- In the starting the key is kept in a register given as K.
- For each round (R), which 31 rounds, the 64-bit key that will be used in this round the most significant 64-bits of the current K.
- After that, the Key will be updating as follows:
    1. 61 times of rotating the key bit positions.
    2. The $k_{19}k_{18}k_{17}k_{16}k_{15}$ bits of K are XORed with the round counter. (first round 1 = 00001 and last round 31 = 11111).
    3. The last four bits are inserted over the S-Box.
- Add-Round-Key: EX-ORing between the block of plaintext (each 4-bit as block) and the K.
- Substitution Layer:
- S-Box is input 4-bit and output 4-bit. o The input is formed as **Table 1** to get output

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 1: S-Box



- Permutation Layer: a mixing bit as **Figure 2**.

Step 1

Key

This key is used in encrypting the PRESENT algorithm because of the great power of this key in establishing cryptographic operations securely. A strong key must be used to ensure that the data is secure and cannot be easily hacked or decrypted. This key was chosen based on considerations of security and efficiency in encrypting data using the PRESENT algorithm.

This key is used in encryption and decryption as it consists of 256 bits

Step 2

encryption

To encrypt a given text using the PRESENT algorithm, you must specify the appropriate key that will be used in the encryption process. Then the data to be encrypted is selected and the PRESENT algorithm is applied to it according to the rules and steps specified in the algorithm. This converts the original data into an encrypted form and a set of steps to restore the original text. The PRESENT algorithm is used in many security and cryptographic applications where good performance and high resistance to attacks are desirable features

Step 3

Decreption

To decrypt ciphertext using the PRESENT algorithm, the following steps can be followed:

1. Select the encrypted data: You must select the ciphertext you wish to decrypt using the PRESENT algorithm.

2. Determine the correct key: The correct key originally used in the encryption process must be determined to ensure correct decryption.

3. Implementing the PRESENT algorithm: The PRESENT algorithm is implemented using encrypted data and the key to get the original text.

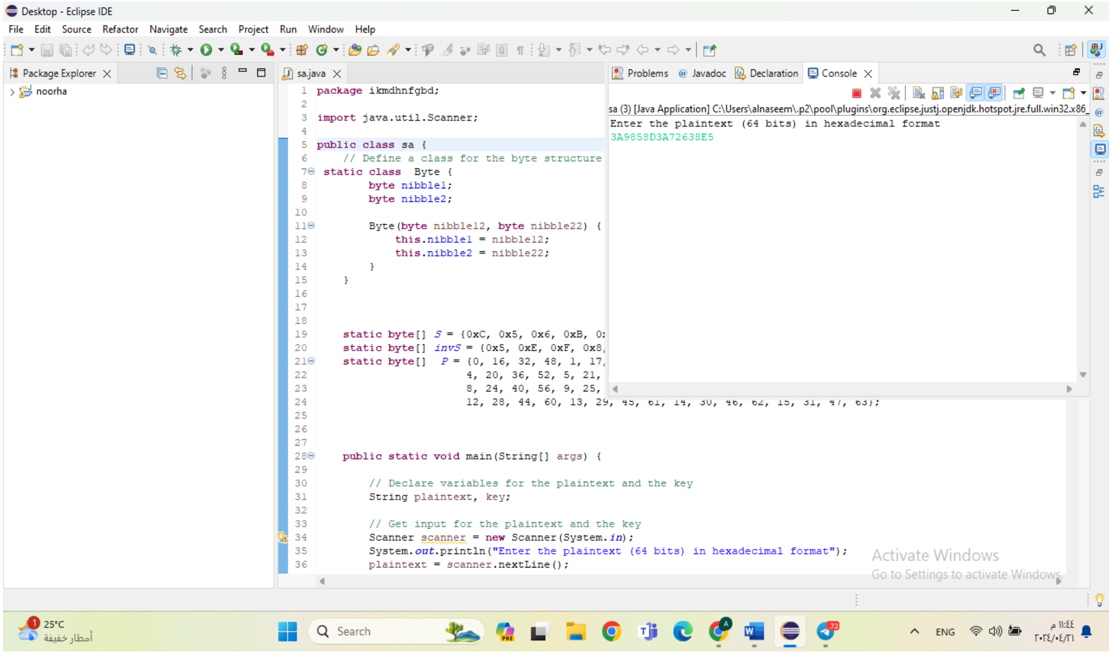
4. Recover the original text: When the process is completed, the original text encrypted in the previous step will be obtained.

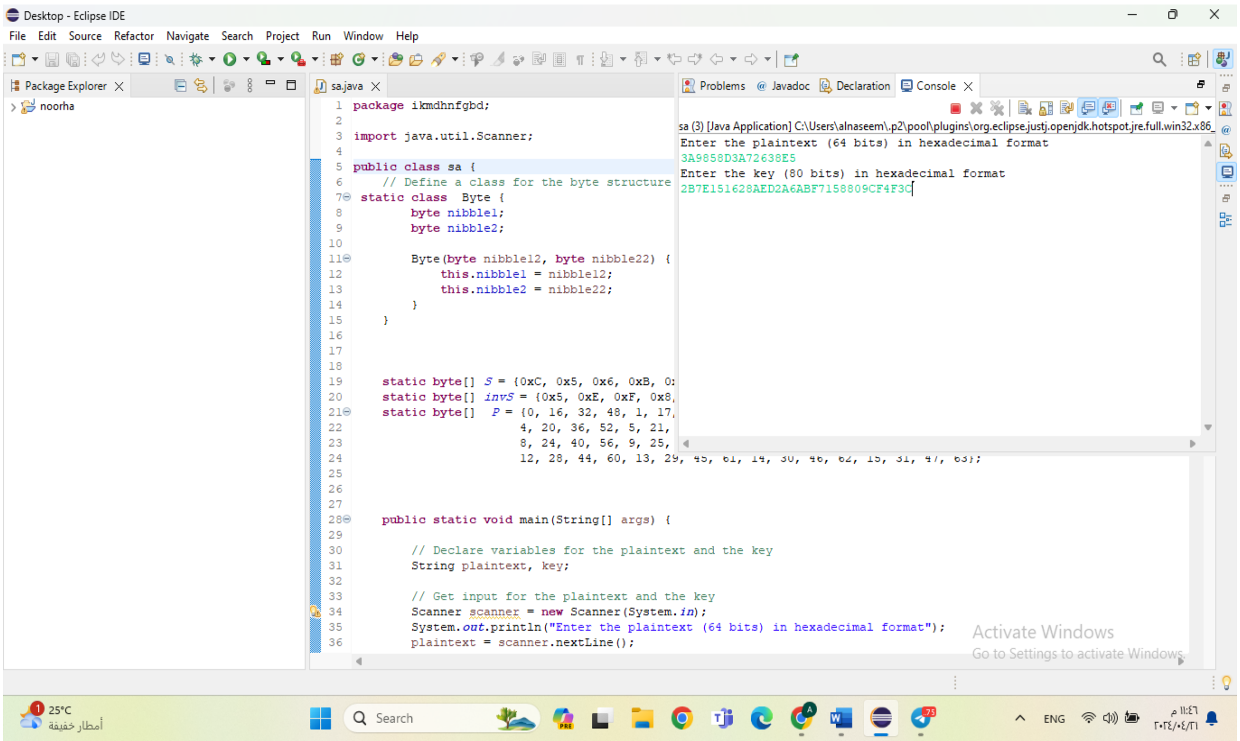
CHAPTER FOUR

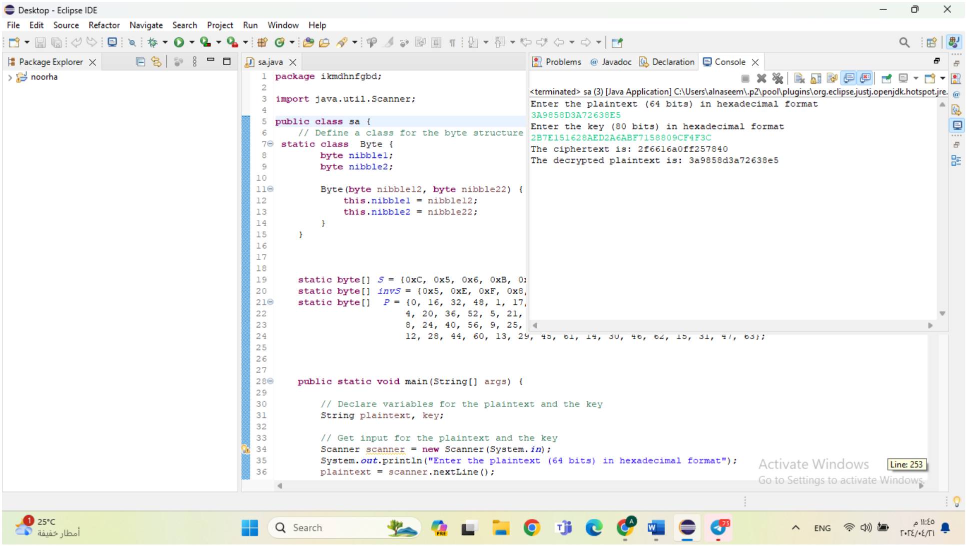
:. 4.1 Introduction

This chapter presents the results that are achieved from executing the project system's steps that described in chapter three. These results are explained in section (4.2). This project produces some conclusions with working on the system are listed in section (4.3). The future works are presented at the end of this chapter in section (4.4) of what the futurity ideas can be adopted by researchers to develop this project and introduces a more benefit to the other students

4.2 Design of project






Desktop - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
noorha
sa.java
package ikmdhnfgbd;
import java.util.Scanner;
public class sa {
// Define a class for the byte structure
static class Byte {
byte nibble1;
byte nibble2;
Byte(byte nibble12, byte nibble22) {
this.nibble1 = nibble12;
this.nibble2 = nibble22;
}
}
static byte[] S = {0xC, 0x5, 0x6, 0xB,
static byte[] invS = {0x5, 0xE, 0xF, 0x8,
static byte[] P = {0, 16, 32, 48, 1, 17,
4, 20, 36, 52, 5, 21,
8, 24, 40, 56, 9, 25,
12, 28, 44, 60, 13, 29, 45, 61, 14, 30, 46, 62, 15, 31, 47, 63};
public static void main(String[] args) {
// Declare variables for the plaintext and the key
String plaintext, key;
// Get input for the plaintext and the key
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the plaintext (64 bits) in hexadecimal format");
plaintext = scanner.nextLine();
Problems Javadoc Declaration Console
<terminated> sa (3) [Java Application] C:\Users\alnaseem\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.
Enter the plaintext (64 bits) in hexadecimal format
3A9858D3A72638E5
Enter the key (80 bits) in hexadecimal format
2B7E151628AED2A6ABF7158809CF4F3C
The ciphertext is: 2f6616a0ff257840
The decrypted plaintext is: 3a9858d3a72638e5
Activate Windows
Go to Settings to activate Windows.
Line: 253
25°C
Search
ENG

4.3 *Conclusion*

By implementing the Present Lightweight Algorithm for encrypting text files, this project aims to enhance data security in modern digital environments. The combination of efficiency, scalability, and robust security offered by PLA makes it a promising solution for safeguarding sensitive information in various applications and industries.

4.4 Future Works Suggestions

It can be expected that developments in encryption technologies and applications may make the use of the PRESENT algorithm in image encryption possible in the future. Future use of image encryption using the PRESENT algorithm could be essential in the context of applications that require protection of sensitive images, such as digital medical imaging or secure transmission of images over the Internet

By using appropriate encryption mechanisms, such as integrating the PRESENT algorithm with existing image encryption techniques, the security and confidentiality of data in images can be improved. This is done by modifying the key generation as well as modifying the number of cycles while maintaining security performance

4.4 References

https://www.java.com/en/

[1] T. Sharma, "Lightweight Encryption Algorithms, Technologies, and Architectures in Internet of Things: A Survey," in *Innovations in Computer Science and Engineering*: Springer, 2020, pp. 341-351.

[2] K. Acharya *et al.*, "Analysis of Cryptographic Algorithms for Network Security," vol. 3, no. 2, pp. 130-135, 2013.

[3] I. Bhardwaj *et al.*, "A review on lightweight cryptography algorithms for data security and authentication in IoTs," in *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*, 2017, pp. 504-509: IEEE.

[4] A. Shah *et al.*, "A survey of lightweight cryptographic algorithms for iot-based applications," in *Smart Innovations in Communication and Computational Sciences*: Springer, 2019, pp. 283-293.

[5] S. Aruna *et al.*, "Lightweight Cryptography Algorithms for IoT Resource-Starving Devices," pp. 139-169, 2020.

[6] T. Eisenbarth *et al.*, "A survey of lightweight-cryptography implementations," vol. 24, no. 6, pp. 522-533, 2007.

[7] C. S. R. CENTER. *Lightweight Cryptography*. Available: https://csrc.nist.gov/projects/lightweight-cryptography

[8] D. Hong *et al.*, "LEA: A 128-bit block cipher for fast encryption on common processors," in *International Workshop on Information Security Applications*, 2013, pp. 3-27: Springer.

[9] A. Bogdanov *et al.*, "PRESENT: An Ultra-Lightweight Block Cipher," Berlin, Heidelberg, 2007, pp. 450-466: Springer Berlin Heidelberg.