CRYPT ANALYSIS & CYBER DEFENCE

22CSB3101A

III YEAR ,I SEMESTER

Academic Year: 2024-2025

KONERU LAKSHMAIAH EDUCATION FOUNDATION

CRYPT ANALYSIS & CYBER DEFENCE(22CSB3101A)

Mode: A

LAB WORKBOOK

STUDENT ID:

STUDENT NAME:

ACADEMIC YEAR: 2024-25

KLEF (Deemed to be University), Vaddeswaram

UNIVERSITY VISION AND MISSION

Vision

To be a Globally Renowned University

Mission

To impart quality higher education and to undertake research and extension with emphasis on application and innovation that cater to the emerging societal needs through all-round development of the students of all sections, enabling them to be globally competitive and socially responsible citizens with intrinsic values.

Weblink: https://www.kluniversity.in/Mission.aspx

Department of Computer Science and Engineering

Program Educational Objectives

1. **Practice engineering in a broad range of industrial, societal and real-world applications.**
2. **Pursue advanced education, research and development, by adapting creative and innovative practices in their professional careers.**
3. **Conduct themselves in a responsible, professional, and ethical manner.**
4. **Participate as leaders in their fields of expertise and in activities that support service and economic development throughout the world**

Web link: https://www.kluniversity.in/cse1/vission.aspx

PROGRAM OUTCOMES		
PO	**Graduate Attributes**	**Program Outcome Description**
1	Program Outcome Description	To impart mathematics, science, & engineering knowledge to develop skills to solve complex engineering problems.
2	Problem Analysis	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	Design/ development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4	Conduct investigations of complex problems	An ability to use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5	Modern tool usage	Ability to create, select and apply appropriate techniques, resources and modern engineering activities, while understanding its limitations.
6	The engineer and society	Ability to apply reasoning and the contextual knowledge to assess social & health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practices.
7	Environment and sustainability	Ability to demonstrate the engineering knowledge to find solutions to contemporary issues by understanding their impact on societal and environmental contexts, towards sustainable development
8	Ethics	An ability to apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.
9	Individual and team work	To inculcate abilities to be able to act as a leader as well as team player effectively in multi-disciplinary settings

10	Communication	To develop oral and written communication skills to articulate the complex engineering activities with the engineering community and society effectively through reports and design documentation, make effective presentations, and give and receive clear instructions
11	Project management and finance	To develop working knowledge and understanding of the engineering and management principles to manage projects in multi-disciplinary environments.
12	Lifelong learning	To inculcate the habit of constant knowledge upgrading habit to meet the ever-changing technology and industry needs.
PROGRAM SPECIFIC OUTCOMES		
PSO1	An ability to design and develop software projects as well as to analyze and test user requirements	
PSO2	Working knowledge on emerging technologies as per the industry requirements	

Table of Contents

1. Consider a char pointer containing the string 'Hello World' and create a program that XOR's each character in this string with 1, then displays the resulting values. 23

2 Consider a char pointer containing the string 'Hello World' and create a program that performs bitwise AND or XOR operations on each character in the string with 127, then display the resulting values 33

23. Implementation of Caesar Cipher ---------- 38.

4. Implementation of Vigenere Cipher. 42

5. Implementation of Affine Cipher 49

6. Implementation of Playfair Substitution Technique 5

7. Implementation of Railfence Transposition Technique 11

8. Implementation of Columnar Transposition Technique 11

9. Implementation of One Time Pad Substitution Technique 19

10.Implementation of Hill Cipher Substitution Technique 27

11.Implementation of SDES Key Generation Algorithm 35

12.Implementation of SDES Encryption Algorithm 35

13. Implementation of AES Key Generation. 44

14.Implementation of Substitute Bytes and Shift Rows operations in AES. 7

15.Implementation of SRC4 Algorithm 15

16.Implementation of LCG Pseudorandom Numbers Generator 6

17.Implementation of Blum Blum Shub Generator 11

18.Implementation of RSA Algorithm 16

19.Implementation of Diffie-Hellman Algorithm 5

20.Implementation of Elgamal Cryptosystem Algorithm 2

21.Implementation of Two Simple Hash Functions 4

22.Implementation a Simple Hash Algorithm (SHA-512) 11

23.Implementation a MD5 Algorithm 11

24.Implementation of Digital Signature Algorithm 7

A.Y. 2024-25 LAB CONTINUOUS EVALUATION

S.No	Date	Experiment Name	Pre-Lab (5 M)	In-Lab (25M)			Post-Lab (10 M)	Viva Voce (5 M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10 M)	Analysis & Inference (10 M)				
1.		Consider a char pointer containing the string 'Hello World' and create a program that XOR's each character in this string with 1, then displays the resulting values								
2.		Consider a char pointer containing the string 'Hello World' and create a program that performs bitwise AND or XOR operations on each character in the string with 127, then display the resulting values.								
3.		Implementation of Caesar Cipher								
4.		Implementation of Vigenère Cipher								
5.		Implementation of Affine Cipher								
6.		Implementation of Playfair Substitution Technique								
7.		Implementation of Rail fence Transposition Technique								
8.		Implementation of Columnar Transposition Technique								
9.		Implementation of One Time Pad Substitution Technique								
10.		Implementation of Hill Cipher Substitution Technique								
11.		Implementation of SDES Key Generation Algorithm								

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
12		Implementation of SDES Encryption Algorithm								
13.		Implementation of AES Key Generation.								
14.		Implementation of Substitute Bytes and Shift Rows operations in AES.								
15.		Implementation of SRC4 Algorithm								
16.		Implementation of LCG Pseudorandom Numbers Generator								
17.		Implementation of Blum Blum Shub Generator								
18.		implementation of RSA Algorithm								
19.		Implementation of Diffie-Hellman Algorithm								
20.		Implementation of Elgamal Cryptosystem Algorithm								
21		Implementation of Two Simple Hash Functions								
22		Implementation a Simple Hash Algorithm (SHA-512)								
23		Implementation a MD5 Algorithm								
24		Implementation of Digital Signature Algorithm								

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

1. **Consider a char pointer containing the string 'Hello World' and create a program that XOR's each character in this string with 1, then displays the resulting values.**

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Python.

Distribution

To implement the program, first, declare a character pointer variable pointing to the string "Hello World". Then, iterate through each character in the string using a loop. For each character, perform a bitwise XOR operation with the value 1 to toggle the least significant bit, effectively flipping it from 0 to 1 or from 1 to 0. Print the result character by character to display the modified string. Finally, don't forget to free the memory allocated for the string. This program demonstrates a simple encryption technique by XORing each character with 1, which can be reversed by XORing again with 1 to restore the original string.

Pre-Requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. How can you convert Python strings to character pointers in C?

2. How does Python handle memory management when dealing with character pointers passed from C?

3. Explain how you would handle null-terminated strings when working with character pointers in Python.

4. How do you handle encoding and decoding issues when dealing with character pointers in Python-C interfacing?

5. Discuss the differences in string handling between Python and languages that heavily rely on character pointers, like C

In-Lab Task:

1. *Consider a string (char pointer with a value Hello World and implement a program that should XOR each character 1 in this string with 0 and displays the result.*

Sol:

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva questions:

- What is Python, and what are some of its key features?
- Differentiate between Python 2.x and Python 3.x versions.
- Explain the differences between lists and tuples in Python.
- What are the advantages of using Python for web development?
- Describe the concept of PEP 8 and its significance in Python programming.

Post Lab Task:

1. Calculate factorial of a given number using recursive approach.

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

2. Consider a string char pointer with a value Hello World and implement a program that should AND or and XOR 2 each character in this string with 127 and display the result.

Date of the Session:___/___/___ Time of the Session:_____to_____

Objective

- To understand the concept of Python.

Distribution

This Python program takes a string "Hello World" and performs a bitwise operation (AND or XOR) on each character with the value 127. It iterates through each character in the string, applying the specified operation to each character individually. If the chosen operation is "AND", it performs a bitwise AND operation with 127, preserving only the bits that are common between the character's ASCII value and 127. If the operation is "XOR", it performs a bitwise XOR operation with 127, toggling the bits that are set in one but not both operands. The result is then displayed, showing the modified string where each character has been transformed according to the selected operation.

Pre-Requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. Discuss the differences between shallow copy and deep copy in Python.

2. Explain the usage of lambda functions in Python with examples.

3. What is a Python virtual environment, and why is it used?

4. Discuss the differences between `==` and `is` operators in Python.

5. Explain the role of `self` in Python class methods.

In-Lab Task:

1. *Consider a string char pointer with a value Hello World and implement a program that should AND or and XOR 2 each character in this string with 127 and display the result***.**

Sol:

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva questions:

- What is a Python module, and how is it different from a package?
- How can you handle exceptions in Python? Provide examples.
- What is the purpose of the `__init__` method in Python classes?
- Explain the differences between `range()` and `xrange()` functions in Python 2.
- What are decorators in Python, and how are they used?

Experiment #		Student ID	
Date		Student Name	

.

Post Lab Task:

1. Write a Python Program to implement whether a string is a palindrome or not

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

3.Implementation of Caesar Cipher.

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of substitution techniques.
- To understand Symmetric Crypto System.

Distribution

The Caesar cipher substitution technique is a simple and historical encryption method that involves shifting each letter in the plaintext by a fixed number of positions down or up the alphabet. For example, with a shift of 3, 'A' would become 'D', 'B' would become 'E', and so forth. This technique operates under the principle of modular arithmetic, where the alphabet wraps around at the end. Though easy to implement and understand, Caesar cipher is highly vulnerable to brute-force attacks due to its limited number of possible keys (26 in the case of the English alphabet) and its lack of strong encryption. Despite its weaknesses, it serves as a fundamental concept in cryptography and can be used in conjunction with other techniques for stronger encryption.

Pre-Requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. What is the Caesar cipher, and how does it work?

2. Explain the process of encrypting and decrypting using the Caesar cipher.

3. What is the significance of the "shift" value in the Caesar cipher?

4. How does the Caesar cipher handle spaces and punctuation in the plaintext?

5. Discuss the security implications of using the Caesar cipher for encryption.

In-Lab Task:

1. Write a program to implement Ceasar Cipher encryption for any given plaintext.

(Hint: As a sample input student need to use his/her name as an input for implementing the program. Key that can be used is 4.

Equation for encryption is C=(P+K) Mod 26

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is the key space of the Caesar cipher, and why is it important?
- Describe the process of breaking the Caesar cipher using brute-force attacks.
- How can frequency analysis be used to attack the Caesar cipher?
- Explain the relationship between the Caesar cipher and modular arithmetic.
- What are some practical applications of the Caesar cipher in today's world, if any?

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write a program to implement Ceasar Cipher decryption for any given plaintext.

(Hint: As a sample input student need to use his/her name as an input for implementing the program. Key that can be used is 4.

Equation for decryption is P=(C-K)Mod 26

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

4. Implementation of Vigenère Cipher.

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of substitution techniques.
- To understand Symmetric Crypto System.

Distribution

The Vigenère cipher is a polyalphabetic substitution cipher that adds an extra layer of complexity to the traditional Caesar cipher by using a keyword or phrase as the key. Named after the French cryptographer Blaise de Vigenère, this technique involves shifting each letter of the plaintext by a variable amount based on the corresponding letter in the keyword. The key is repeated cyclically to match the length of the plaintext. Unlike the Caesar cipher, where each letter is shifted by a fixed amount, the Vigenère cipher uses different shifts at different positions in the plaintext, making it more resistant to frequency analysis. Despite its historical use and initial security, the Vigenère cipher can be broken with modern cryptanalysis techniques, especially if the length of the keyword is known. Nonetheless, it serves as a significant milestone in the development of cryptography and laid the groundwork for more sophisticated encryption methods.

Pre-Requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. What is the Vigenère cipher, and how does it differ from the Caesar cipher?

2. Explain the process of encryption using the Vigenère cipher.2.

3. How does the Vigenère cipher use a keyword or phrase as its key?

4. Describe the key space of the Vigenère cipher and its significance in encryption.

5. What is the advantage of using the Vigenère cipher over the Caesar cipher?

In-Lab Task:

1. Write a program to implement Vigenère Cipher encryption for any given plaintext.
(Hint: As a sample input student need to use his/her name as an input for implementing the program. Given key is LEG.
Equation for encryption is C=(P+K) Mod 26

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva questions:

- How does the Vigenère cipher achieve polyalphabetic substitution, and why is it more resistant to frequency analysis?
- Discuss the process of decrypting a message encrypted with the Vigenère cipher.
- What are some weaknesses of the Vigenère cipher, and how can they be exploited in cryptanalysis?
- How does the length of the keyword affect the security of the Vigenère cipher?
- Compare the security and efficiency of the Vigenère cipher with modern encryption algorithms.

Post Lab Task:

1. Write a program to implement Vigenère Cipher decryption for any given plaintext.
(Hint: As a sample input student need to use his/her name as an input for implementing the program. Given key is LEG. Equation for encryption is P=(C-K) Mod 26

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

5. Implementation of Affine Cipher

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand Substitution Techniques
- To understand the applications of Affine Cipher.

Description:

The Affine cipher is a type of monoalphabetic substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. It combines the Caesar cipher's shift with multiplication and addition operations. In the Affine cipher, the encryption function can be represented as E(x) = (ax + b) mod m, where 'a' and 'b' are the keys of the cipher, 'm' is the size of the alphabet, and 'x' is the numerical value of the plaintext letter. To decrypt, one applies the inverse function D(x) = a^(-1)(x - b) mod m, where 'a^(-1)' denotes the modular multiplicative inverse of 'a' modulo 'm'. The Affine cipher is more secure than the Caesar cipher but still susceptible to frequency analysis. It operates on the principle of modular arithmetic and requires two keys for encryption and decryption.

Pre requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. What is the Affine cipher, and how does it differ from the Caesar cipher?

2. Explain the mathematical formula used in the encryption and decryption process of the Affine cipher.

3. How does the Affine cipher achieve encryption and decryption using modular arithmetic?

4. What are the requirements for selecting valid keys in the Affine cipher?

5. Discuss the significance of the keys 'a' and 'b' in the Affine cipher and how they affect the encryption process.

In-Lab Task:

1. Write a program to implement Affine Cipher encryption for any given plaintext.
(Hint: As a sample input student need to use his/her name as an input for implementing the program. Given a=4; b=7. Equation for Encryption is E=(ax+b) mod m.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- How does the choice of alphabet size 'm' affect the security and complexity of the Affine cipher?
- Describe the process of encrypting and decrypting a message using the Affine cipher with a specific set of keys.
- What are the advantages of using the Affine cipher over simpler substitution ciphers like the Caesar cipher?
- Discuss the vulnerabilities of the Affine cipher and potential attacks to break the encryption.
- Compare the security of the Affine cipher with other classical and modern encryption techniques.
-

Experiment #		Student ID	
Date		Student Name	

Post-Lab Task:

1. Write a program to implement Affine Cipher encryption for any given plaintext.
(Hint: As a sample input student need to use his/her name as an input for implementing the program. Given a=4; b=7. Equation for Encryption is $D=a^{-1}(x-b) \bmod m$.

Experiment #		Student ID	
Date		Student Name	

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A

CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

6. Implementation of Playfair Cipher

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of multiple-letter encryption.
- To understand the applications of the technique.

Description:

The Playfair Cipher is a polygraphic substitution cipher that encrypts pairs of letters instead of single letters. It uses a 5x5 matrix (usually called a Playfair Square) containing a keyword to encrypt and decrypt the plaintext. Each letter is replaced by the letter in the same row and column of the pair it forms with the neighboring letter. If the letters are in the same row or column, they are replaced by the next letter in that row or column, forming a rectangle.

Pre requisites:

Experimental Setup:

- Programming Language: Python 3.9
- Integrated Development Environment (IDE): Jupyter Notebook
- Libraries Used: None

Pre-Lab Task:

1. Define diagram with an example.

2. What is the reason to consider a 5 × 5 matrix in a playfair cipher technique?

3. What to do if letters in plaintext reoccur eg: *Hello*?

4. What are the advantages of Playfair cipher?

5. Trace what will be the encrypted message by using Playfair cipher if the message is‘balloon’ and the key is “Monarchy”.

In-Lab Task:

1) Write a code to implement encryption for Playfair Cipher Substitution Technique for the following input:Sample Input:-

Plain Text: “Student to consider his/her name”

Secret Key: REDHATCLUB

R	E	D	H	A
T	C	L	U	B
F	G	I/J	K	M
N	O	P	Q	S
V	W	X	Y	Z

*(**Note**: Ignore the whitespace and consider the text)*

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Briefly explain the Playfair Cipher and its basic principles.
- How are the Playfair Cipher's key matrix and key phrase related?
- What steps are involved in encrypting a message using the Playfair Cipher?
- How is the Playfair Cipher decrypted?
- Discuss any limitations or weaknesses of the Playfair Cipher.

Post-Lab Task:

1) Write a code to implement decryption for Playfair Cipher Substitution Technique. Consider the ciphertext for the corresponding plaintext given in In-lab task,.

Secret Key: REDHATCLUB

R	E	D	H	A
T	C	L	U	B
F	G	I/J	K	M
N	O	P	Q	S
V	W	X	Y	Z

Sol)

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(for Evaluators Use Only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

7. Implementation of Rail fence Transposition Techniques

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of Railfence Transposition techniques

Description

The Rail Fence transposition technique is a simple form of transposition cipher where the plaintext is written diagonally in a zigzag pattern across multiple rails (rows) of an imaginary fence. To encrypt, characters of the plaintext are written in a zigzag pattern across a specified number of rails. Once the entire message is written out, the ciphertext is formed by reading off the letters row by row from the rails. Decryption involves reconstructing the rails and reading off the characters in the same zigzag pattern to retrieve the original plaintext. The number of rails used determines the complexity and periodicity of the pattern, affecting both the security and readability of the ciphertext. While historically significant and easy to understand, the Rail Fence cipher is susceptible to frequency analysis and becomes less effective with longer messages or fewer rails.

Pre requisites:

Experimental Setup:
Programming Language: Python 3.9

Integrated Development Environment (IDE): Jupyter Notebook

Libraries Used: None

Pre-Lab Task:

1. What is transposition cipher?

2. What are the applications of rail-fence cipher?

3. Brief description of columnar transposition cipher.

4. Columnar transposition cipher is also known as_____.

5. Difference between substitution and transposition techniques?

In-Lab Task:

1. Write a program to implement Rail-Fence Cipher (encryption) for any given plaintext.

Sample input: - N=6 Message=S21location56

Sample output: - Si2to1anac5to6l

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is the Rail Fence transposition technique, and how does it work?
- Describe the process of encrypting a message using the Rail Fence cipher.
- Explain the steps involved in decrypting a message encrypted with the Rail Fence cipher.
- What are some practical applications of the Rail Fence cipher in modern cryptography?
- Compare the Rail Fence cipher with other transposition ciphers in terms of security and efficiency.

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write a program to implement Rail fence Cipher (decryption) for any given plaintext.

Sample input: - N=6

Ciphertext= Si2to1anac5to6l

Sample output: S21location56

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

8. Implementation of Columnar Transposition Techniques

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of Columnar techniques

Description

The columnar transposition technique is a form of transposition cipher where the plaintext is written into a grid of columns and then read out row by row to create the ciphertext. To encrypt, the plaintext message is written into a grid with a specified number of columns. The key for this cipher is the order in which columns are read to create the ciphertext. For example, if the key specifies column order 2, 1, 4, 3, then the second column is read first, followed by the first column, and so on. Decryption involves rearranging the ciphertext into the original grid using the inverse of the column order specified by the key and then reading out the rows to retrieve the plaintext. The security of the columnar transposition cipher relies on the strength of the key, which determines the column order. While offering more complexity compared to simpler transposition ciphers like the Rail Fence, it can still be vulnerable to frequency analysis and other cryptanalytic attacks if the key is weak or known.

Prerequisites:

Experimental Setup:
Programming Language: Python 3.9

Integrated Development Environment (IDE): Jupyter Notebook

Libraries Used: None

Pre-Lab Task:

1. What is the columnar transposition technique, and how does it differ from substitution ciphers?

2. Explain the process of encrypting a message using the columnar transposition cipher.

3. How does the key affect the encryption and decryption processes in the columnar transposition cipher?

4. Describe the steps involved in decrypting a message encrypted with the columnar transposition cipher.

5. Discuss the role of the column order key in the security of the columnar transposition cipher.

In-Lab Task:

1. Write a program to implement Columnar Cipher (encryption) for a given plaintext.

Sample input: - HACKATHONKLU Key 3 1 2 4

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Can you illustrate the encryption and decryption processes of a message using the columnar transposition cipher with a specific key?
- Compare the security strengths and weaknesses of the columnar transposition cipher with other classical ciphers.
- What are some practical applications of the columnar transposition cipher in modern cryptography?
- How does the columnar transposition cipher handle spaces, punctuation, and non-alphabetic characters in the plaintext?
- What are some potential attacks or vulnerabilities of the columnar transposition cipher, and how can they be mitigated?

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write a program to implement Columnar Cipher (decryption) for a given plaintext. Consider ciphertext for the given plaintext HACKATHON KLU in In-lab task.

Sample input: - Key 2 3 1 4

Experiment #		Student ID	
Date		Student Name	

(For Evaluators Use Only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: out of Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

9. Implementation of One Time Pad Substitution Techniques

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of One -Time Pad Substitution technique

Description

The one-time pad is a theoretically perfect symmetric encryption technique where each plaintext bit is XORed with a corresponding bit from a truly random key of the same length, often generated as a random sequence of bits. This technique ensures that the ciphertext is statistically random and provides perfect secrecy if the key is used only once, hence the name "one-time pad". Due to the key's randomness and length matching the plaintext, every possible plaintext of that length is equally probable, making cryptanalysis impossible without knowledge of the key. However, maintaining the security of the key distribution is crucial, as any repetition or reuse of the key undermines the security of the encryption. One-time pads are impractical for most everyday encryption due to the challenges of securely distributing and managing truly random keys of sufficient length, but they serve as a benchmark for evaluating the security of other encryption schemes.

Pre requisites

Experimental Setup:
Programming Language: Python 3.9

Integrated Development Environment (IDE): Jupyter Notebook

Libraries Used: None

Pre-Lab Task:

1. What is the one-time pad encryption technique?

2. What are the key characteristics of a one-time pad?

3. Why is the one-time pad considered a perfect encryption technique?

4. What are the main vulnerabilities or limitations of the one-time pad?

5. How does the security of the one-time pad compare to other encryption techniques?

In-Lab Task:

1. Write a program to implement One time Pad Cipher (encryption) for a given plaintext.

Sample input: - HELLO Key XMCKL

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Can the one-time pad be vulnerable to known-plaintext attacks or chosen-plaintext attacks?
- What are some practical applications or historical uses of the one-time pad?
- What are some examples of errors in implementing the one-time pad that could compromise its security?
- How does the XOR operation play a crucial role in the one-time pad encryption?
- What are some modern alternatives to the one-time pad that address its limitations?

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write a program to implement One time Pad Cipher (decryption) for a given plaintext.

Sample input: - EQNVZ Key XMCKL

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured: out of Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

10. Implementation of Hill Substitution Techniques

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand the concept of Encryption and Decryption.
- To understand the applications of Hill Cipher Substitution technique

Description

The Hill cipher is a polyalphabetic substitution cipher based on linear algebra, developed by Lester S. Hill in 1929. Unlike simple substitution ciphers that replace each letter individually, the Hill cipher operates on groups of letters (typically pairs or triplets), treating them as vectors over a finite field. Encryption involves multiplying the plaintext vector by a matrix (the encryption key) modulo the size of the alphabet. Decryption requires multiplying the ciphertext vector by the matrix's inverse, also modulo the alphabet size. The Hill cipher provides a higher level of security than monoalphabetic substitution ciphers due to its polyalphabetic nature and the complexity introduced by matrix operations. However, it requires careful key management, including ensuring the key matrix is invertible and its size matches the plaintext vector length, making it vulnerable to frequency analysis and attacks if not implemented correctly.

Prerequisites:

Experimental Setup:
Programming Language: Python 3.9

Integrated Development Environment (IDE): Jupyter Notebook

Libraries Used: None

Pre-Lab Task:

1. What is the basic principle behind the Hill cipher?

2. How does the key matrix affect encryption and decryption in the Hill cipher?

3. What are the advantages of the Hill cipher over monoalphabetic substitution ciphers?

3. What are the key requirements for a key matrix in the Hill cipher?

4. How does the Hill cipher handle spaces and punctuation in plaintext?

In-Lab Task:

1. Write a program to implement Hill Cipher (encryption) for a given plaintext.

Input : Plaintext: ACT
Key: GYBNQKURP
Output : Ciphertext: POH

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What are the vulnerabilities of the Hill cipher?
- Can the Hill cipher be used for encryption of digital data like images or files?
- How does the Hill cipher compare in terms of computational complexity with modern encryption algorithms like AES?
- What are some practical applications of the Hill cipher today?
- What improvements or modifications can be made to enhance the security of the Hill cipher?

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write a program to implement Hill Cipher (decryption) for a given plaintext.

Input : Ciphertext: POH
Key:IFKVIVVMI
Output : Plaintext: ACT

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured: out of Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

11. Implementation of SDES Key Generation Algorithm

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand symmetric key block cipher technique.
- To generate keys using SDES Key Generation Algorithm

Description:

A simplified data encryption key generation algorithm in cryptography typically involves several fundamental steps to ensure secure and efficient key management. Firstly, a random number generator is utilized to generate a sequence of bits that form the basis of the encryption key. This sequence is then processed through a key derivation function, which transforms it into a format suitable for encryption algorithms while enhancing its entropy and security. The resulting key undergoes validation to ensure it meets cryptographic standards and then gets distributed securely to authorized parties. Throughout this process, emphasis is placed on randomness, entropy, and protection against cryptographic attacks to guarantee the confidentiality and integrity of encrypted data.

Pre requests:

1. Programming Language: Choose a programming language of your preference to implement the DES algorithm. Some commonly used languages for cryptography include Python, Java, C++, or C#. Ensure that the language supports cryptographic libraries or modules for efficient implementation.

2. Cryptographic Libraries: Depending on the programming language you choose, you might need to use cryptographic libraries or modules that provide functions and methods for encryption, decryption, and other cryptographic operations. Examples include:

- Python: PyCryptodome, cryptography
- Java: Java Cryptography Architecture (JCA) or Bouncy Castle
- C++: Crypto++ or OpenSSL
- C#: .NET Framework's Cryptography API or Bouncy Castle

3. IDE or Text Editor: You'll need an integrated development environment (IDE) or a text editor to write and run your code. Some popular options are:

- Python: PyCharm, Visual Studio Code, or Jupyter Notebook

- Java: Eclipse, IntelliJ IDEA, or Visual Studio Code
- C++: Visual Studio, Code::Blocks, or CLion
- C#: Visual Studio, Visual Studio Code, or JetBrains Rider

4. Test Data: You'll need test data for encrypting and decrypting using the DES algorithm. Prepare a set of sample data that you can use to verify the correctness of your implementation.

5. DES Algorithm Specification: Obtain the DES algorithm specification, which includes the step-by-step instructions and rules for encryption and decryption. This specification will guide your implementation.

Pre-Lab Task:

1. What is the key length used in the DES algorithm?

2. How many bits are used for the actual encryption key in DES, and how many for parity

3. Describe the initial step in the DES key generation algorithm.

4. What is the purpose of the Permuted Choice 1 (PC-1) step in DES key generation?

5. How many subkeys are generated from the initial key in DES, and what is the size of each subkey?

In-Lab:

1. **Write a program to implement Simpleified DES Key Generation Algorithm. Consider the given input**

Key (10 – bits) : 1 1 0 1 0 0 0 0 0 1

P_{10} order: *3 5 2 7 4 10 1 9 8 6*

P_8 order: *6 3 7 4 8 5 10 9*

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions

- Explain the process of rotating and permuting the key during key schedule generation.
- What role does the Permuted Choice 2 (PC-2) step play in DES key schedule generation?
- How many rounds of key scheduling are there in DES, and why is this number significant?
- Discuss the importance of the key schedule in relation to the security of the DES algorithm
- How does the key schedule ensure that each round of DES encryption uses a different subkey?

Experiment #		Student ID	
Date		Student Name	

Post Lab:

1. Write a Pseudocode to implement SDES key generation algorithm.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPTANALYSIS AND CYBER DEFENSE WORKBOOK

12. Implementation of SDES Encryption Algorithm

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To understand symmetric key block cipher technique.
- To encrypt given plaintext using SDES Algorithm

Description:

A simplified data encryption algorithm in cryptography involves transforming plaintext into ciphertext using a key, ensuring secure communication and data confidentiality. Typically, such algorithms consist of a series of systematic steps: first, the plaintext is segmented into fixed-size blocks; next, each block undergoes substitution and permutation based on the encryption key. This process, known as encryption, obscures the original content, making it unreadable without the corresponding decryption key. Modern algorithms, such as Advanced Encryption Standard (AES), utilize complex mathematical operations like substitution boxes, permutations, and key expansion to achieve strong encryption. The encrypted ciphertext is then securely transmitted or stored, safeguarding sensitive information from unauthorized access or interception.

Pre requests:

6. Programming Language: Choose a programming language of your preference to implement the DES algorithm. Some commonly used languages for cryptography include Python, Java, C++, or C#. Ensure that the language supports cryptographic libraries or modules for efficient implementation.

7. Cryptographic Libraries: Depending on the programming language you choose, you might need to use cryptographic libraries or modules that provide functions and methods for encryption, decryption, and other cryptographic operations. Examples include:

- Python: PyCryptodome, cryptography
- Java: Java Cryptography Architecture (JCA) or Bouncy Castle
- C++: Crypto++ or OpenSSL
- C#: .NET Framework's Cryptography API or Bouncy Castle

8. IDE or Text Editor: You'll need an integrated development environment (IDE) or a text editor to write and run your code. Some popular options are:

- Python: PyCharm, Visual Studio Code, or Jupyter Notebook
- Java: Eclipse, IntelliJ IDEA, or Visual Studio Code

- C++: Visual Studio, Code::Blocks, or CLion
- C#: Visual Studio, Visual Studio Code, or JetBrains Rider

9. Test Data: You'll need test data for encrypting and decrypting using the DES algorithm. Prepare a set of sample data that you can use to verify the correctness of your implementation.

10. DES Algorithm Specification: Obtain the DES algorithm specification, which includes the step-by-step instructions and rules for encryption and decryption. This specification will guide your implementation.

Pre-Lab Task:

1. What is the block size used in the DES algorithm, and why is this size significant?

2. Describe the structure of the DES algorithm, including its key length and number of rounds

3. Explain the importance of initial and final permutations in the DES algorithm.

4. How does DES achieve confusion and diffusion in its encryption process?

5. What role do S-boxes play in the DES algorithm, and how many S-boxes are used?

In-Lab Task:

1. Write a program to implement SDES Encryption Algorithm using the given inputs

Plain Text (8 - bits): Use the binary equivalent of the last Number of your register number.

Test case: If the last number is 0, then consider the predecessor non-zero value.

Initial Permutation: 2 6 3 1 4 8 5 7

Expanded Permutation: 4 1 2 3 2 3 4 1

$$S0 = \begin{matrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{matrix} \qquad S1 = \begin{matrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{matrix}$$

P_4: 2 4 3 1

Key (10 – bits) : 1 1 0 1 0 0 0 0 0 1

P_{10} order: *3 5 2 7 4 10 1 9 8 6*

P_8 order: *6 3 7 4 8 5 10 9*

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions

- Explain the process of rotating and permuting the key during key schedule generation.
- What role does the Permuted Choice 2 (PC-2) step play in DES key schedule generation?
- How many rounds of key scheduling are there in DES, and why is this number significant?
- Discuss the importance of the key schedule in relation to the security of the DES algorithm
- How does the key schedule ensure that each round of DES encryption uses a different subkey?

Experiment #		Student ID	
Date		Student Name	

Post Lab:

1. Write a Pseudocode to implement DES key generation algorithm.

Experiment #		Student ID	
Date		Student Name	

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

13. Implementation of AES Key Generation.

Date of the Session:___/___/____ **Time of the Session:_____to_____**

Objective

- You will understand the importance of generating a strong and unpredictable key for AES encryption.
- You will learn about the key length options for AES and their impact on security.
- AES supports three key lengths: 128-bit, 192-bit, and 256-bit. The longer the key length, the stronger the encryption, but it may also increase computational overhead.
- You will gain an understanding of the need for randomness in key generation.
- You may explore techniques for deriving a key from a passphrase or password.

Description

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used for securing sensitive data. AES key generation involves the process of generating a suitable encryption key for use with the AES algorithm. The key is generated according to specific rules and requirements defined by the AES standard.

Pre requisites:

1. OpenSSL: OpenSSL is a widely used open-source cryptographic library that provides various functions, including AES key generation. You can use OpenSSL command-line tools or programming interfaces to generate AES keys.

2. CryptGenRandom (Windows): CryptGenRandom is a Windows API function that generates cryptographically secure random numbers. You can utilize this function in your programming language of choice (such as C/C++, C#, or PowerShell) to generate AES keys.

3. Cryptography libraries: Various programming languages offer cryptography libraries that include functions for AES key generation. For example, Python has libraries like `cryptography` and `pycryptodome` that provide methods for generating AES keys.

4. Hashcat: Hashcat is a popular password recovery and cracking tool that supports various encryption algorithms, including AES. While it is primarily used for password cracking, it can also generate AES keys.

5. Java Cryptography Architecture (JCA): If you are working with Java, you can utilize the Java Cryptography Architecture, which provides APIs for generating AES keys. The `javax.crypto.KeyGenerator` class can be used to generate AES keys of different sizes.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1) How many keys are used in AES with respective to key size? And how many rounds are there in AES?

2) What is AES?

3) How is the key generated in AES?

4) What are the steps involved in AES key generation?

5) : What are the characteristics of a good AES key?

Experiment #		Student ID	
Date		Student Name	

In Lab:

1. Write a program to implement the AES Key generation

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Explain the importance of key generation in the AES algorithm.
- How is the key expansion process performed in AES?
- Describe the different key sizes supported by AES and their implications.
- What is the role of the S-box in AES key generation?
- Discuss any security considerations or best practices regarding AES key generation.

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write Pseudocode for AES Key Generation Algorithm.

Experiment #		Student ID	
Date		Student Name	

(for Evaluators Use Only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

14.Implementation of Substitute Bytes and Shift Rows operations in AES.

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To use S-Box to perform a byte-by-byte substitution of the block.
- To understand shift row operation using permutation.

Description:

Substitute Bytes and Shift Rows are two operations used in the AES algorithm as part of the encryption process. Substitute Bytes involves replacing each byte in a block of data with a corresponding byte from a substitution box. Shift Rows involves shifting the rows of the data block cyclically to the left by different offsets.

Pre requisites:

1. PyCryptodome: PyCryptodome is a powerful library that provides cryptographic functions, including AES, in Python. It offers a high-level interface for implementing AES operations. You can use the `AES` module from PyCryptodome to perform Substitute Bytes and Shift Rows operations.

2. OpenSSL: OpenSSL is a widely used open-source library that implements cryptographic functions, including AES. It provides a command-line tool called `openssl` that allows you to perform cryptographic operations. You can use the `openssl` command-line tool with appropriate options to perform Substitute Bytes and Shift Rows operations.

3. Java Cryptography Architecture (JCA): If you prefer to work with Java, you can utilize the Java Cryptography Architecture (JCA) to implement AES operations. The `javax.crypto` package in Java provides classes and interfaces for cryptographic operations. You can use the `Cipher` class from this package to perform AES encryption/decryption, including the Substitute Bytes and Shift Rows operations.

4. Cryptography.io: Cryptography.io is a Python library that provides a simple and easy-to-use API for various cryptographic operations, including AES. You can use the `cryptography . hazmat. primitives. ciphers` module from this library to perform AES operations. It allows you to implement Substitute Bytes and Shift Rows operations as part of your AES implementation.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1) How many keys are used in AES with respective to key size? And how many rounds are there in AES?

2) Why AES is better than DES, Double DES, Triple DES?

3) How many rounds will take place if it 128bit, 192bit, and 256bit respectively?

4) Name the methods present in AES transformation function.

In-Lab Task:

1. Siri is a part of cryptanalysis team in an organization. The team is developing a complete application to decrypt a message using AES algorithm, the team lead has given 'Shift Rows Module' to Siri. To complete this module Siri asked you to write a program which performs shift rows operation in AES algorithm. Write the code by following the Input and Output Format given below:

[Hint: The first row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. The following is an example of Shift Rows]. 4*4 input and output matrices are given below:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions :

- What is the purpose of the Substitute Bytes operation in AES?
- How does the Substitute Bytes operation achieve confusion in the AES encryptionprocess?
- Explain the process of performing the Shift Rows operation in AES.
- How does the Shift Rows operation provide diffusion in the AES encryption process?
- Can you explain the importance of the Substitute Bytes and Shift Rows operations in AESin terms of security?

Post Lab Task:

1. Write a program to implement Substitute Bytes operation in AES Encryption Algorithm. Consider the given input and output. Use Substitute Byte Table

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

15.Implementation of SRC4 Algorithm

Date of the Session:___/___/___ **Time of the Session:_____to_____**

Objective

- To generate keys.
- To encrypt the given Plaintext

Description:

The SRC4 (Simplified Rivest Cipher 4) algorithm is a simplified yet widely used stream cipher in cryptography, primarily known for its efficiency and ease of implementation. This keystream is then combined with the plaintext using bitwise XOR operation to produce the ciphertext. SRC4 is notable for its simplicity in key scheduling and encryption process: it initializes a permutation of all possible byte values (0 to 7) based on the key, then iteratively modifies this permutation during encryption. Despite its widespread use in the past, SRC4 is no longer considered secure against modern cryptographic attacks due to vulnerabilities in its key scheduling algorithm, leading to widespread abandonment in favor of more robust ciphers like AES (Advanced Encryption Standard).

Pre requisites:

Programming Language: Python 3.9

Integrated Development Environment (IDE): Jupyter Notebook

Pre-Lab Task:

1. What does RC4 stand for, and who developed it?

2. Describe the key length range typically used in RC4 encryption. Why is this range significant?

3. Explain the basic principle behind the RC4 algorithm. How does it generate the keystream?

4. What is the significance of the RC4 initialization phase?

5. How does RC4 use the generated keystream to encrypt plaintext?

In-Lab Task:

1.Write a program to implement SRC4 Initial Permutation of S.

Input: S = [0 1 2 3 4 5 6 7]; T = [1 2 3 6 1 2 3 6]; P = [1 2 2 2]; K = [1 2 3 6]

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Discuss the vulnerabilities that led to the decline in RC4's security reputation.
- What measures can be taken to mitigate known vulnerabilities in RC4?
- Compare and contrast RC4 with block ciphers like AES in terms of encryption approach and security features.
- Explain the impact of weak keys on the security of RC4. How are weak keys identified and managed?
- What are some practical applications where RC4 might still be used despite its vulnerabilities?

Post Lab Task:

1. Write a program to implement SRC4 Encryption Algorithm.

Input: S = [0 1 2 3 4 5 6 7]; T = [1 2 3 6 1 2 3 6]; P = [1 2 2 2]; K = [1 2 3 6]

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluators Use Only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

16. Implementation of LCG Pseudorandom Numbers Generator

Date of the Session:___/___/___ Time of the Session: ____to_____

Objective

- To understand and implement Pseudo random number generation.
- To generate secure pseudorandom generators using LCG.

Description:

A linear congruential pseudorandom number generator (LCG) in cryptography is a fundamental algorithmic approach used to produce sequences of numbers that appear random. It operates on the principle of iterating through a sequence of numbers using a linear equation $Xn+1=(a\cdot Xn+c) \bmod m$, where X_n is the current pseudorandom number, is a multiplier, c is an increment, and m is the modulus. The security and quality of an LCG depend heavily on the choice of parameters; poorly chosen parameters can lead to predictable sequences and vulnerabilities in cryptographic applications. Despite its simplicity, LCGs are vulnerable to attacks such as the correlation attack and should be used cautiously in cryptographic contexts, often supplemented with additional techniques or replaced by more sophisticated generators like cryptographic-strength pseudorandom number generators (CSPRNGs) to ensure robust security.

Pre requisites:

1. Python:

- You can use the built-in `random` module in Python for basic random number generation, including LCG.

- For implementing BBS, you can utilize the `pycryptodome` library, which provides cryptographic functionalities, including a BBS implementation.

2. Java:

- Java provides the `java.util.Random` class for basic random number generation, including LCG.

- For BBS implementation, you can use cryptographic libraries such as Bouncy Castle, which offers cryptographic algorithms and functions.

3. C/C++:

- C and C++ do not have built-in random number generation capabilities, but you can use the `rand()` function from the standard library for LCG.

- For BBS implementation, you can use cryptographic libraries like OpenSSL or Crypto++.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1. What is a Linear Congruential Generator (LCG) and how does it generate pseudorandom numbers?

2. What are the key parameters of an LCG and how do they affect the randomness and security of the generated sequence?

3. Why is the choice of modulus m important in an LCG?

4. What are the typical characteristics of the sequence generated by an LCG?

5. How can an LCG be attacked or exploited in a cryptographic context?

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1.Write a program to implement Linear Congruential Pseudorandom number Generator (LCG) .

Input: m=9; a=2; c=0.

Viva Questions:

- What is a Linear Congruential Generator (LCG), and how does it work?
- What are the key parameters involved in an LCG?
- What are the common attacks against LCGs when used in cryptography?
- How can the period of an LCG impact its security in cryptography?
- What are the potential applications of LCG?

Post Lab Task:

1. Write Pseudocode for Linear Congruential Pseudo Random Generator.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

17. Implementation of Blum Blum Shub Generator

Date of the Session:___/___/____ Time of the Session: _____to_____

Objective

- To understand and implement Pseudo random number generation.
- To generate secure pseudorandom generators using BBS.

Description:

The Blum Blum Shub (BBS) generator is a cryptographic pseudorandom number generator (CPRNG) based on number theory, specifically designed to provide strong security guarantees. It operates by utilizing the quadratic residuosity problem in modular arithmetic. The generator's key parameters are three large prime numbers, p, q, and an initial seed x0x_0x0, which must be chosen carefully to ensure cryptographic security. The output sequence of the BBS generator is derived from successive squaring of Xn modulo N=p× where N is the modulus. The security of BBS relies on the difficulty of factoring NNN into its prime components, ensuring that predicting future outputs without knowledge of p and q remains computationally infeasible. BBS is notable for its theoretical robustness and has been utilized in various cryptographic applications where strong randomness and security are paramount**.**

Pre requisites:

3. Python:

- You can use the built-in `random` module in Python for basic random number generation, including LCG.
- For implementing BBS, you can utilize the `pycryptodome` library, which provides cryptographic functionalities, including a BBS implementation.

4. Java:

- Java provides the `java.util.Random` class for basic random number generation, including LCG.
- For BBS implementation, you can use cryptographic libraries such as Bouncy Castle, which offers cryptographic algorithms and functions.

3. C/C++:

- C and C++ do not have built-in random number generation capabilities, but you can use the `rand()` function from the standard library for LCG.
- For BBS implementation, you can use cryptographic libraries like OpenSSL or Crypto++.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1. What is the fundamental principle behind the Blum Blum Shub (BBS) generator?

2. What are the key parameters of the BBS generator, and why are they crucial for security?

3. How does the BBS generator ensure cryptographic security?

4. What are the main advantages of using the BBS generator in cryptography?

5. Discuss the potential weaknesses or vulnerabilities of the BBS generator.

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1.Write a program to implement Blum Blum Shub Pseudorandom number Generator (BBS).

Input: p=11;q-23;s=3

Viva Questions:

- How is the security of the BBS generator influenced by the size of the primes ppp and qqq?
- What are some practical applications of the BBS generator in cryptography?
- What are the computational requirements for generating BBS pseudorandom numbers?
- How can the output of the BBS generator be tested for randomness and quality?
- In what scenarios would you recommend using the BBS generator over other PRNGs in cryptography?

Post Lab Task:

1. Write Pseudocode for Blum Blum Shub Pseudorandom Number Generator (BBS).

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

18.Implementation of RSA Algorithm

Date of the Session:___/___/_____ **Time of the Session: _to_____**

Objective

- To implement RSA algorithm.
- To understand encryption as a block.

Description:

The RSA (Rivest-Shamir-Adleman) algorithm is a widely used public-key encryption system. In this experiment, participants will implement the RSA algorithm from scratch. They will understand the mathematical concepts behind RSA, including modular arithmetic, prime number generation, and the Chinese Remainder Theorem. They will code the key generation process, encryption, and decryption algorithms. The participants will also experiment with different key sizes and analyze the trade-offs between security and computational efficiency.

Pre requisites:

Implementing RSA requires both programming skills and mathematical understanding. Here are some commonly used software tools and programming languages that can be used to complete the implementation of RSA:

1. Programming Languages:

- Python: Python is a popular choice for implementing RSA due to its simplicity and extensive library support. The `cryptography` library in Python provides built-in functions for RSA key generation, encryption, and decryption.

- Java: Java also has libraries, such as `javax.crypto`, that offer cryptographic functions, including RSA.

2. OpenSSL:

- OpenSSL is a widely-used open-source software library that provides cryptographic functions. It includes RSA key generation, encryption, and decryption functions. OpenSSL is available for various programming languages, including C, C++, Python, and Java.

3. Cryptography Libraries:

- There are several cryptographic libraries available that provide RSA implementations.

Some commonly used ones include:

Bouncy Castle: A comprehensive cryptography library for Java and C#, which includes RSA functionality.

- Crypto++: A C++ library that provides a wide range of cryptographic algorithms, including RSA.

- Cryptography.io: A Python library that offers a high-level interface for cryptographic operations, including RSA.

4. Integrated Development Environments (IDEs):

- IDEs can greatly assist in developing and testing RSA implementations. Some popular IDEs for different programming languages include:

- PyCharm: A Python IDE that offers advanced code editing, debugging, and testing features.

- Eclipse: An open-source IDE for Java development, which provides a range of plugins for cryptographic libraries.

- Visual Studio Code: A versatile code editor that supports multiple programming languages, including Python and Java.

Pre-Lab Task:

1. What is the RSA algorithm, and what problem does it solve in cryptography?

2. What are the key components of the RSA algorithm?

3. How does RSA ensure confidentiality in communication?

4. What is the significance of prime numbers in RSA?

5. How are RSA keys generated, and what considerations are important for key size?

In-Lab Task:

1. Write a program to implement RSA algorithm.

   Input:

   p = 3 and q = 11.

   n = p * q = 3 * 11 = 33.

   φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20.

   Choose e such that 1 < e < φ(n) and e and φ (n) are coprime. ...e=7

   Compute a value for d such that (d * e) % φ(n) = 1. ...d=33

   Public key is (e, n) => (7, 33)

   Private key is (d, n) => (3, 33)

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Describe the key generation process in RSA.
- How does the RSA encryption process work?
- What is the role of the Euler's totient function in RSA?
- Explain the RSA decryption process.
- Discuss the security strengths and limitations of RSA.

Experiment #		Student ID	
Date		Student Name	

Post Lab Task:

1. Write Pseudocode for RSA Asymmetric Encryption Algorithm.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

19. Implementation of Diffie-Hellman Algorithm

Date of the Session:___/___/____ **Time of the Session: _____ to_____**

Objective

- To understand the key exchange process.
- To understand the purpose of Discrete Logarithms.

Description:

The Diffie-Hellman algorithm is a key exchange protocol that allows two parties to establish a shared secret key over an insecure channel. In this experiment, participants will implement the Diffie-Hellman algorithm. They will understand the mathematical principles behind the algorithm, including modular exponentiation and discrete logarithms. Participants will code the key generation process, the exchange of public keys, and the derivation of the shared secret key. They will verify that the shared key is the same for both parties.

Pre requests :

There are several software libraries and tools available for implementing the Diffie-Hellman algorithm. Here are some popular options:

1. OpenSSL: OpenSSL is a widely used open-source library that provides cryptographic functions, including support for the Diffie-Hellman key exchange. It is available for multiple programming languages, including C/C++ and Python.

2. Cryptography.io: Cryptography.io is a Python library that provides various cryptographic algorithms, including Diffie-Hellman. It offers an easy-to-use API for generating and exchanging Diffie-Hellman keys.

3. Bouncy Castle: Bouncy Castle is a comprehensive cryptography library available for Java and C#. It includes support for Diffie-Hellman key exchange and various other cryptographic algorithms.

4. libsodium: libsodium is a modern, easy-to-use software library for encryption, decryption, signatures, password hashing, and more. It provides a simple API for implementing Diffie-

Hellman in several programming languages, including C/C++, Python, and JavaScript.

5. NaCl (Networking and Cryptography library): NaCl is a high-level cryptographic library that aims to be easy to use and resistant to misuse. It includes support for Diffie-Hellman key exchange and various other cryptographic operations. NaCl is available for several programming languages, including C/C++, Python, and JavaScript.

Pre-Lab Task:

1. What is the difference between RSA and Diffie Hellman?

2.What are the main properties of Diffie Hellman?

3.Explain Asymmetric key cryptography in few words and draw a neat diagram on its working.

4.Explain the concept of key exchange in the Diffie-Hellman algorithm.

5.How does the Diffie-Hellman algorithm enable secure communication over an insecure channel?

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1. You are trying to encrypt your messages you want to send to your friend because you don't want an outsider to know the confidential information you are sending to your friend so in order to do that use Diffie-Hellman Algorithm to encrypt the messages(choose an appropriate example) and encrypt the messages using Diffie-Hellman Algorithm.

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- Describe the key exchange process in the Diffie-Hellman algorithm.
- What is the role of the primitive root modulo in Diffie-Hellman?
- How does the Diffie-Hellman algorithm ensure secure key exchange over an insecure channel?
- Explain the process of computing the shared secret key in Diffie-Hellman.
- Discuss the potential vulnerabilities or attacks on the Diffie-Hellman algorithm.

Post-Lab Task:

1. Write Pseudocode for Diffie-Hellman Key Exchange algorithm.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

20. Implementation of Elgamal Cryptosystem Algorithm

Date of the Session:___/___/____ **Time of the Session: ____to_____**

Objective: The objective of this experiment is to implement the Elgamal cryptosystem, a public-key encryption algorithm, and evaluate its security and efficiency in protecting sensitive information.

Description: Elgamal is based on the Diffie-Hellman key exchange protocol and provides secure communication between two participants using public-private key pairs. The experiment entails implementing the Elgamal algorithm's key generation, encryption, and decryption processes. Key generation requires the selection of suitable prime numbers and the calculation of the required parameters. Encryption involves converting plaintext into ciphertext using the recipient's public key, whereas decryption utilizes the recipient's private key to recover the plaintext message.

In addition to evaluating the efficacy of the Elgamal algorithm in terms of computational complexity and encryption/decryption speed, the experiment will also assess its computational complexity and encryption/decryption speed. In addition, the security of the Elgamal cryptosystem will be evaluated in lightof probable attacks, including brute-force, chosen-plaintext, and known-plaintext attacks. The findings will shed light on the advantages and disadvantages of the Elgamal algorithm in practical cryptographic applications.

Pre requisities:

1. Programming Languages:

- Python: A versatile language with several cryptographic libraries.
- Java: Provides built-in support for cryptographic operations.
- C/C++: Offers low-level control and high-performance capabilities.

2. Cryptographic Libraries:

- Cryptography (Python): A powerful library that provides various cryptographic algorithms, including ElGamal.
- Bouncy Castle (Java/C#): A comprehensive library offering cryptographic algorithms and protocols.
- OpenSSL (C/C++): A widely used library that provides a range of cryptographic functions.

3. Mathematical Libraries:

- GMP (GNU Multiple Precision Arithmetic Library): A library for high-precision arithmetic operations, useful for large number computations involved in ElGamal.
- BigInteger (Java): A class in Java's standard library for handling arbitrary precision integers.

4. Integrated Development Environments (IDEs):

- PyCharm (Python): A popular IDE for Python development.
- Eclipse (Java): An IDE widely used for Java development.
- Visual Studio (C/C++): A powerful IDE for C/C++ development.

Remember that implementing cryptographic algorithms is a complex task, and it's crucial to follow best practices and guidelines to ensure security. It is advisable to consult cryptographic experts and references while implementing such algorithms. Additionally, make sure to thoroughly test your implementation and consider using well-established libraries whenever possible to avoid common pitfalls and vulnerabilities.

Pre Lab

1. What is the ElGamal cryptosystem?

2. How does the ElGamal cryptosystem work?

3. What are the advantages of the ElGamal cryptosystem?

4.What are the limitations of the ElGamal cryptosystem?

5. How can the security of the ElGamal cryptosystem be enhanced?

In Lab:

1. Write a program to implement the Elgamal Cryptosystem Algorithm.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is the Elgamal cryptosystem used for?
- Explain the key generation process in the Elgamal cryptosystem.
- How does the Elgamal algorithm achieve secure communication?
- Can you discuss the mathematical principles behind the Elgamal cryptosystem?
- What are the advantages and limitations of the Elgamal cryptosystem?

Post Lab Task:

1. Write Pseudocode for Elgamal Algorithm

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

21. Implementation of Two Simple Hash Functions

Date of the Session:___/___/___ Time of the Session: ____to_____

Objective

- Understanding the concept of hashing
- Apply the Basic hash function implementation
- Apply the Hash distribution
- Apply the Collision resolution
- Apply the Efficiency considerations

Description:

In this experiment, participants will implement two simple hash functions. They will understand the basic principles of hash functions, such as input compression and the avalanche effect. Participants will code the hash functions using a programming language of their choice and test them with various inputs. They will evaluate the collision resistance and distribution properties of the hash functions and discuss their limitations.

Pre requisities:

There are several software tools and programming languages you can use to implement two hash functions. Here are a few commonly used options:

1. Python: Python is a versatile programming language with a rich set of libraries and modules that make it suitable for hash function implementation. You can use the built-in hashlib module to implement common hash functions like MD5, SHA-1, SHA-256, etc. Additionally, you can also implement custom hash functions using Python's standard library.

2. C/C++: C and C++ are low-level programming languages that provide greater control over memory and hardware. You can implement hash functions from scratch or use existing libraries like OpenSSL to perform hashing operations.

3. Java: Java provides built-in libraries such as java .security. Message Digest that support various hash functions like MD5, SHA-1, SHA-256, etc. You can use these libraries to implement hash functions in your Java application.

4. JavaScript: JavaScript has built-in support for hash functions through the Crypto API. You can use functions like `crypto.createHash` or `crypto.subtle.digest` to implement hash functions in the browser or Node.js environment.

5. Ruby: Ruby has libraries like Digest that provide implementations of various hash functions. You can use them to calculate hashes easily in your Ruby application.

6. Go: Go programming language offers the crypto package that includes hash functions. You can import and use functions like sha256.New() or md5.New() to implement hash functions in Go.

7. Rust: Rust is a systems programming language that prioritizes memory safety and performance. You can use libraries like Rust Crypto's `digest` crate to implement hash functions in Rust.

Pre-Lab Task:

1. How does the simple addition hash function work?

2. What are the advantages of the simple addition hash function?

3. What are the limitations of the simple addition hash function?

1. How does the simple XOR hash function work?

2. What are the advantages of the simple XOR hash function?

In-Lab Task:

1. Write a program to implement Simple Addition Hash Function.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is a hash function, and what are its primary characteristics?
- Describe the implementation and working principles of the two simple hash functions.
- Discuss the collision resistance property of hash functions.
- How can the quality of a hash function be evaluated?
- Explain any potential weaknesses or limitations of the implemented hash functions.

Post Lab Task:

1. Write a program to implement Simple XOR Hash Function.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

22. Implementation a Simple Hash Algorithm (SHA-512)

Date of the Session:___/___/____ **Time of the Session: ____to_____**

Objective

- To understand the importance of Hash function for secure data transmission.
- To implement SHA-512 hash algorithms.

Description :

SHA-512 (Secure Hash Algorithm 512-bit) is a widely used cryptographic hash function. In this experiment, participants will implement the SHA-512 algorithm. They will understand the internal workings of the algorithm, including message padding, message expansion, and the compression function. Participants will code the SHA-512 algorithm and verify its correctness by comparing the output with existing implementations. They will also discuss the security properties of SHA-512, such as pre-image resistance and collision resistance.

Pre requests:

To implement a simple hash algorithm like SHA-512, you can use various programming languages and libraries. Here are a few examples of software tools you can use:

1. Python:
   - hashlib module: The hashlib module in Python provides various hash algorithms, including SHA-512. You can import the module and use the `sha512()` function to compute the hash.
2. Java:
   - Java Cryptography Architecture (JCA): The JCA provides a set of cryptographic APIs in Java. You can use the `MessageDigest` class to compute the SHA-512 hash.
3. C/C++:
   - OpenSSL library: OpenSSL is a widely used open-source library that provides cryptographic functions. It includes an implementation of SHA-512.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1. What is SHA Algorithm?

2. What is the hash value of SHA-1?

3. Write any 3 differences between SHA-1 and SHA-256?

4. What is SHA256 hash function?

5. Can you explain the steps involved in the SHA-512 algorithm?

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1. Kiran is doing an internship in Cyber Security. As part of his research, he is assigned a task to demonstrate the working of SHA-512 algorithm in Computer programming language.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is the purpose of the SHA-512 algorithm?
- How does the SHA-512 algorithm differ from other hash algorithms?
- How is the integrity of data ensured using SHA-512?
- What are some applications of SHA-512 in cryptography and security?
- How does SHA-512 ensure data integrity and message authentication?

Post Lab Task:

1. Write Pseudocode for Secure Hash Algorithm (SHA-512).

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	Marks Secured:_____ out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

23. Implementation of Message Digest (MD5) Algorithm

Date of the Session:___/___/____ **Time of the Session: ____to_____**

Objective

- To understand the importance of Hash function for secure data transmission.
- To implement MD5 hash algorithm.

Description :

MD5 (Message Digest Algorithm 5) is a widely used cryptographic hash function designed to produce a 128-bit hash value, typically represented as a 32-character hexadecimal number, from any input data. Developed by Ronald Rivest in 1991, MD5 operates by repeatedly processing blocks of data through a series of cryptographic functions to generate a unique fixed-size hash, which ideally changes significantly even with minor alterations to the input data. While historically popular for digital signatures and checksums due to its speed and efficiency, MD5 is now considered cryptographically broken, with vulnerabilities that can lead to collisions (different inputs producing the same hash) being exploited in practical attacks, making it unsuitable for most security applications today.

Pre requests:

To implement a simple hash algorithm like MD5, you can use various programming languages and libraries. Here are a few examples of software tools you can use:

3. Python:

- hashlib module: The hashlib module in Python provides various hash algorithms, including SHA-512. You can import the module and use the ‘MD5()` function to compute the hash.

4. Java:

- Java Cryptography Architecture (JCA): The JCA provides a set of cryptographic APIs in Java. You can use the `MessageDigest` class to compute the MD5 hash.

3. C/C++:

- OpenSSL library: OpenSSL is a widely used open-source library that provides cryptographic functions. It includes an implementation of MD5.

Experiment #		Student ID	
Date		Student Name	

Pre-Lab Task:

1. Why was MD5 widely used in the past for cryptographic applications?

2. What are the main vulnerabilities associated with MD5 today?

3. Explain the concept of a 'collision' in the context of MD5.

4.How does MD5 compare to other cryptographic hash functions like SHA-256 in terms of security?

5 What are some practical applications where MD5 might still be used despite its vulnerabilities?

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1. Kiran is doing an internship in Cyber Security. As part of his research, he is assigned a task to demonstrate the working of MD5 algorithm in Computer programming language.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What is MD5, and what does it stand for?
- Who developed the MD5 algorithm, and when was it introduced??
- Explain the basic structure and operation of the MD5 algorithm.
- What are the primary applications of MD5 in cryptography?
- Discuss the security properties of MD5. Is it still considered secure today?

Post Lab Task:

1. Write Pseudocode for Message Digest (MD5) Algorithm.

Experiment #		Student ID	
Date		Student Name	

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SUBJECT CODE: 22CSB3101A
CRYPT ANALYSIS AND CYBER DEFENCE WORKBOOK

24. Implementation of Digital Signature Algorithm

Date of the Session:___/___/___

Objective

- Understand the basics of hash functions
- Apply Digital Signature for a given message.

Description:

The Digital Signature Algorithm (DSA) is a cryptographic algorithm used for creating and verifying digital signatures, which are essential in ensuring the authenticity, integrity, and non-repudiation of digital messages or documents. DSA operates within the framework of public-key cryptography, relying on the mathematical properties of discrete logarithms in finite fields. Key components of DSA include a private key for signing messages and a corresponding public key for verifying signatures. To create a digital signature, DSA uses a hash function to generate a digest of the message, which is then processed using the private key to produce a signature. Verifying the signature involves using the signer's public key, along with the original message and the signature itself, to confirm the authenticity and integrity of the message. DSA provides a robust method for ensuring secure communication and transaction validation across various digital platforms and applications.

Pre requisities:

1. What is digital Signature?

2. How does the Digital Signature Algorithm (DSA) differ from other digital signature schemes?

3: What are the key components of the Digital Signature Algorithm (DSA)?

4: Explain the process of generating a digital signature using DSA.

5. How does DSA ensure the security and integrity of digital signatures?

Experiment #		Student ID	
Date		Student Name	

In-Lab Task:

1.Durga is doing an internship in Cyber Security. As part of his research, he is assigned a task to demonstrate the working of Digital Signature algorithm in Computer programming language.

Experiment #		Student ID	
Date		Student Name	

Viva Questions:

- What are the security considerations when choosing parameters for DSA?
- What is the role of hashing algorithms in the Digital Signature Algorithm (DSA)?
- How does DSA handle the issue of message padding?
- What are some common attacks against digital signatures, and how does DSA mitigate these risks?
- In what scenarios is DSA commonly used, and why might it be preferred over other digital signature algorithms??

3. **Post-Lab Task:**

1. Write Pseudocode for Digital Signature Algorithm.

Experiment #		Student ID	
Date		Student Name	

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation

LAB WORKBOOK

(DEEMED TO BE UNIVERSITY)

KONERU LAKSHMAIAH EDUCATION FOUNDATION,
GREEN FIELDS, VADDESWARAM,
GUNTUR-522502

www.kluniversity.in

CATEGORY 1 UNIVERSITY
BY MHRD, Govt. of India

KL ACCREDITED BY NAAC WITH A++ GRADE

nirf 2023 NATIONAL INSTITUTIONAL RANKING FRAMEWORK

RANKED 28 AMONG ALL UNIVERSITIES

43 YEARS OF EDUCATIONAL LEADERSHIP