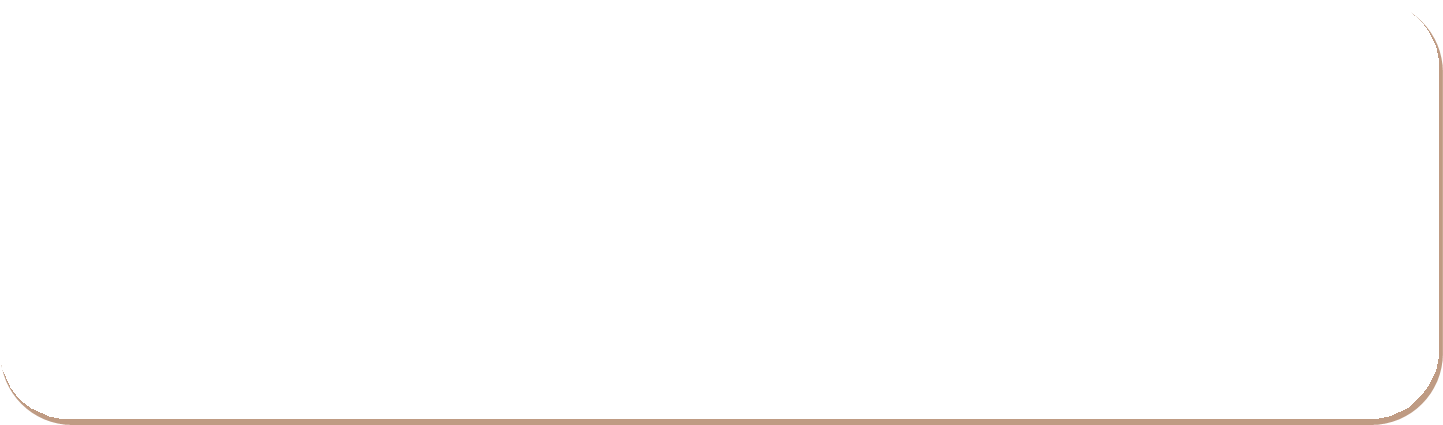
**Laboratory Manual**

## DEPARTMENT OF COMPUTER ENGINEERING AND TECHNOLOGY



**PROGRAM - B.E.**

**CRYPTOGRAPHY AND NETWORK SECURITY LAB (20CSC31)**

**(Common for CSE, CSE-IoT)**

**SEMESTER - VII**

**R-20 Regulation**

***Prepared by:* P. Vimala Manohara Ruth *Verified by:* Prof. N. Rama Devi**



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution, Affiliated to Osmania University | Approved by AICTE |

Accredited by NAAC with A++ Grade | NBA Accredited for six Programs)

Chaitanya Bharathi (PO), Kokapet (V), Gandipet (M), Hyderabad– 500 075, Telangana, India

www.cbit.ac.in



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution, Affiliated to Osmania University | Approved by AICTE |

Accredited by NAAC with A++ Grade | NBA Accredited for six Programs)

Chaitanya Bharathi (PO), Kokapet (V), Gandipet (M), Hyderabad– 500 075, Telangana, India

www.cbit.ac.in

**DEPARTMENT OF COMPUTER ENGINEERING & TECHNOLOGY**

|  |
| --- |
| ***Vision of Institute***  To be a Centre of Excellence in Technical Education and Research |
| ***Mission of Institute***  To address the emerging needs through quality technical education and advanced research |
| ***Quality Policy***  CBIT imparts value based Technical Education and Training to meet the requirements of students, Industry, Trade/ Profession, Research and Development Organizations for Self-sustained growth of Society. |
| ***Vision of the Department***  To be in the frontiers of Computer Science and Engineering with academic excellence and Research. |
| ***Mission of the Department***  The mission of the Computer Engineering and Technology Department is to:   1. Educate students with the best practices of Computer Science by integrating the latest research into the curriculum 2. Develop professionals with sound knowledge in theory and practice of Computer Science and Engineering 3. Facilitate the development of academia-industry collaboration and societal outreach programs 4. Prepare students for full and ethical participation in a diverse society and encourage lifelong learning |



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution, Affiliated to Osmania University | Approved by AICTE |

Accredited by NAAC with A++ Grade | NBA Accredited for six Programs)

Chaitanya Bharathi (PO), Kokapet (V), Gandipet (M), Hyderabad– 500 075, Telangana, India

www.cbit.ac.in

**DEPARTMENT OF COMPUTER ENGINEERING & TECHNOLOGY**

|  |
| --- |
| ***Program Educational Objectives (PEOs)***  PEO 1: Graduates will apply their knowledge and skills to succeed in their careers and/or obtain advanced degrees, provide solutions as entrepreneurs.  PEO 2: Graduates will creatively solve problems, communicate effectively, and successfully function in multi-disciplinary teams with superior work ethics and values.  PEO 3: Graduates will apply principles and practices of Computer Science, Mathematics and Science to successfully complete hardware and/or software-related engineering projects to meet customer business objectives.  PEO 4: Graduates will have the ability to adapt, contribute, innovates modern technologies and systems in the domain of Cyber Security, IoT or productively engage in research. |
| ***Program Outcomes (POs)***  1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of 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**: 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**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.  **6. The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.  **7. Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.  **8. Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.  **9. Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.  **10. Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.  **11. Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.  **12. Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |
| ***Program Specific Outcomes (PSOs)***  PSO 1: Able to acquire the practical competency through emerging technologies and open- source platforms related to the areas of Cyber Security, IoT and Block Chain.  PSO 2: Able to assess the hardware and software aspects necessary for the development of solutions to secure critical IT infrastructure and prepare collaborative plans for any incidence response.  PSO 3: Able to provide diversified solutions in product development by adhering to ethical values for the benefit of society. |



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution, Affiliated to Osmania University | Approved by AICTE |

Accredited by NAAC with A++ Grade | NBA Accredited for six Programs)

Chaitanya Bharathi (PO), Kokapet (V), Gandipet (M), Hyderabad– 500 075, Telangana, India

www.cbit.ac.in

**DEPARTMENT OF COMPUTER ENGINEERING & TECHNOLOGY**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Name of the Laboratory Course with Code:***  CRYPTOGRAPHY AND NETWORK SECURITY LAB(20CSC31) | | | | | | | | | | | | |
| ***Course Outcomes (COs):*** On, 1.  CO1. Identify basic security attacks and services.  CO2. Design symmetric and asymmetric key algorithms for cryptography.  CO3. Create and use of Authentication functions.  CO4. Identify and investigate network security threat.  CO5. Analyze and design network security protocols. | | | | | | | | | | | | |
| ***Course Articulation Matrix:*** | | | | | | | | |  |  |  |  |
| ***CO*** | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| CO1 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO2 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO3 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO4 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO5 |  |  |  |  |  |  |  |  |  |  |  |  |
| Avg |  |  |  |  |  |  |  |  |  |  |  |  |



**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution, Affiliated to Osmania University | Approved by AICTE |

Accredited by NAAC with A++ Grade | NBA Accredited for six Programs)

Chaitanya Bharathi (PO), Kokapet (V), Gandipet (M), Hyderabad– 500 075, Telangana, India

www.cbit.ac.in

**DEPARTMENT OF COMPUTER ENGINEERING & TECHNOLOGY**

CRYPTOGRAPHY AND NETWORK SECURITY LAB (20CSC31)

**INDEX**

|  |  |
| --- | --- |
| **Item** | **Page No** |
| Institute Vision, Institute Mission, Quality Policy, Department Vision, Department Mission and PEO's |  |
| Program Outcomes (POs) |  |
| Program Specific Outcomes (PSOs) |  |
| Syllabus |  |
| Course Introduction |  |
| Assessment Procedure |  |
| General Instructions for Laboratory Classes |  |

**LABORATORY / PRACTICAL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ExpNo** | **Laboratory / Practical** | **CO** | **BTL** | **Page No** |
| 1 | Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program should XOR each character in this string with 0 and displays the result. |  |  |  |
| 2 | Write a C program that contains a string (char pointer) with a value  ‘Hello world’. The program should AND or and XOR each character in this string with 127 and display the result. |  |  |  |
| 3 | Write a Java program to perform encryption and decryption using the following algorithms  a. Ceaser cipher b. Substitution cipher c. Hill Cipher d. Play fair Cipher |  |  |  |
| 4 | Write a C/JAVA program to implement the DES algorithm logic. |  |  |  |
| 5 | Write a C/JAVA program to implement the Blowfish algorithm logic. |  |  |  |
| 6 | Write a C/JAVA program to implement the Rijndael algorithm logic. |  |  |  |
| 7 | Write the RC4 logic in Java Using Java cryptography; encrypt the text  “Hello world” using Blowfish. Create your own key using Java key tool. |  |  |  |
| 8 | Write a Java program to implement RSA algorithm. |  |  |  |
| 9 | Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. |  |  |  |
| 10 | Calculate the message digest of a text using the SHA-1 algorithm in JAVA. |  |  |  |
| 11 | Calculate the message digest of a text using the MD5 algorithm in JAVA. |  |  |  |
| 12 | Implement Simple Columner Transposition technique and Advanced Columner Transposition technique. |  |  |  |
| 13 | Implement Euclidean Algorithm and Advanced Euclidean Algorithm. |  |  |  |
| 14 | Familiarize the cryptographic tools (opencv). |  |  |  |
| **Experiments Beyond curriculum (Open ended / Structured enquiry)** | | | | |
| 15 | Generate Key using One-Time pad to perform encryption and decryption using vernam cipher. |  |  |  |

**20CSC31**

**CRYPTOGRAPHY AND NETWORK SECURITY LAB**

**L T P C**

**0- 0- 2- 1**

Instruction 2 Hours per week

Duration of End Examination 3 Hours

Semester End Examination 50 Marks

Continuous Internal Evaluation 50 Marks

Credits 1

**Course Objectives:** The objectives of this course are to

1. To provide practical understanding of cryptography and its application to network security.

2. To learn various approaches on encryption techniques, strengths of Traffic Confidentiality, Message Authentication Codes.

3. To familiarize with symmetric and asymmetric cryptography.

4. Able to understand the significant functionalities of secure communication.

**Course Outcomes:** On successful completion of the course, students will be able to

1. Identify basic security attacks and services.

2. Design symmetric and asymmetric key algorithms for cryptography.

3. Create and use of Authentication functions.

4. Identify and investigate network security threat.

5. Analyze and design network security protocols

**List of Experiments:**

1. Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program should XOR each character in this string with 0 and displays the result.

2. Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program should AND or and XOR each character in this string with 127 and display the result.

3. Write a Java program to perform encryption and decryption using the following algorithms

a. Ceaser cipher

b. Substitution cipher

c. Hill Cipher

d. Play fair Cipher

4. Write a C/JAVA program to implement the DES algorithm logic.

5. Write a C/JAVA program to implement the Blowfish algorithm logic.

6. Write a C/JAVA program to implement the Rijndael algorithm logic.

7. Write the RC4 logic in Java Using Java cryptography; encrypt the text “Hello world” using Blowfish. Create your own key using Java key tool.

8. Write a Java program to implement RSA algorithm.

9. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript.

10. Calculate the message digest of a text using the SHA-1 algorithm in JAVA.

11. Calculate the message digest of a text using the MD5 algorithm in JAVA.

12. Implement Simple Columner Transposition technique and Advanced Columner Transposition technique.

13. Implement Euclidean Algorithm and Advanced Euclidean Algorithm.

14. Familiarize the cryptographic tools (opencv).

**Text Books:**

1. William Stallings, “Cryptography and Network Security: Principles and Practice” Pearson Education, 6th Edition.

2. Chris Brenton, “Mastering Network Security” Bk&Cd-Rom Edition 2017

**Suggested Reading:**

1. J.W. Rittiaghouse and William M.Hancok “Cyber Security Operations Handbook” Elseviers.

2. Eric Chou, “Mastering Python Networking” 3rd Edition, 2020.

3. Jean-Philippe Aumasson “Serious Cryptography: A Practical Introduction to Modern Encryption”, 2017.

**Online Resources:**

1. https://onlinecourses.nptel.ac.in/noc21\_cs16/preview

**COURSE INTRODUCTION**

**CRYPTOGRAPHY AND NETWORK SECURITY LAB (20CSC31)**

## What is Cryptography?

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit.

Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

**Basic Concepts**

**Cryptography**

The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form.

**Plaintext**

The original intelligible message.

**Cipher text**

The transformed message.

**Cipher**

An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods.

**Key**

Some critical information used by the cipher, known only to the sender& receiver.

**Encipher (encode)**

The process of converting plaintext to cipher text using a cipher and a key.

**Decipher (decode)**

The process of converting cipher text back into plaintext using a cipher and akey.

**Cryptography**

Study of encryption principles/methods.

**Cryptanalysis**

The study of principles and methods of transforming an unintelligible message back into an intelligible message without knowledge of the key. It is also called as code breaking.

**Cryptology**

Both cryptography and cryptanalysis.

**Code**

An algorithm for transforming an intelligible message into an unintelligible one using a code-book.

**ASSESSMENT PROCEDURE AND AWARD OF CIE MARKS**

Following is the subdivision for the internal marks (50) of the Lab:

1. 20 marks for the lab internal tests

Two tests are to be conducted i.e one test after 1st cycle of experiments and second test after the second cycle. Average of two tests marks put together should be consider (20 maximum)

1. 30 marks for CIE

For the CIE 30 marks will be awarded based on the rubrics provided below.

This Rubrics are general guideline. Based on the lab type (programming or hardware) and complexity of the course Rubrics can be customized by the department in tune to program and course offered. Performance Indicators of the Rubrics also can be changed by the departments/Program based on the need.

**RUBRICS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.no | Performance Indicator | Equal or above 80% | Equal or above 60% to less than 80% | Equal or above 40% to less than 60% |
| 1 | Pre Experiment Preparation work –  5 M | Well prepared for the experimentation with a clear specifications, plan/design and additional information | Adequately prepared for the experimentation with clear specifications and plan/design | Minimal preparation and without clear specifications and plan/design |
| 2 | Experimentation (Problem solving, Methodology of Conduction) 10 M | Student conducts experiment with all possible test cases, | Student solves the problem with all possible test cases | Student solves the problem with few test cases, |
| 3 | Post Experiment Analysis[Viva, Inference]  5M | Demonstrates the simulation/ findings /Hardware results Infers and answer all the Questions posed by Instructor | Demonstrates Partial results and inference;  Able to answer Few Questions posed by Instructor | Demonstrates Partial results and inference;  Unable to answer the Questions posed by Instructor |
| 4 | Report Writing  5M | Report with well-organized content, visuals, graphics, citations and references | Student report is as per the format and specifications | Student report is incomplete and inadequate |
| 5 | Conduct (Ethics, Safety, Team Work)  5M | Excellent team spirit, strictly follows ethics and safety precautions | Follows the safety precautions, practices ethics and poor team work | Follows safety precautions and ethical practices |
| Total Score | | | |  |

## ASSESSMENT PROCEDURE AND AWARD OF SEE MARKS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Performance Indicator** | Equal or above 80% | Equal or above 60% to less than 80% | Equal or above 40% to less than 60% |
| 1 | Record  5 M | The content of all the experiments is well-organized, recorded the tables and neatly drawn graphs. Results and discussions are well presented. | The content of most of the experiments are well-organized, recorded the tables, neatly drawn graphs. Results and discussion are presented. | Most of the experiments are incomplete and inadequate. |
| 2 | Write up about the experiment  10M | Presentation of the given experiment/program is very well organized with the required content, specifications, plan/procedure of the conduct and all the required additional information. | Presentation of the given experiment/program is organized with the required content, specifications, plan/procedure of the conduct. | Presentation of the given experiment/program is minimal and is without clear specifications and plan/design. |
| 3 | Conduction of the experiment and observations  15M | Conducts experiment / Simulate the problem with proper connections / all possible test cases. All the possible observations are noted. | Conducts experiment / Simulate the problem with connections / with most of the test cases. Most of the possible observations are noted. | Conducts experiment / Simulate the problem with improper connections / less test cases. |
| 4 | Results & Analysis  10M | Demonstrates the experimental results with adequate analysis / simulation/ findings /obtained results /plotting the graphs. | Demonstrates the experimental results with required analysis / simulation/ findings /obtained results /plotting the graphs. | Demonstrates the partial experimental results with least analysis / simulation/ findings. |
| 5 | Viva-Voce  10M | Answers most of the questions with good analytical explanation. | Answers most of the questions with good explanation. | Answers only few of the questions with nominal explanation. |
| Total Score | | | |  |

**GENERAL INSTRUCTIONS FOR LABORATORY CLASSES**

**DO‘S**

1. Without Prior permission do not enter into the Laboratory.
2. While entering into the LAB students should wear their ID cards.
3. The Students should come with proper uniform.
4. Students should sign in the LOGIN REGISTER before entering into the laboratory.
5. Students should come with observation and record note book to the laboratory.
6. Students should maintain silence inside the laboratory.
7. After completing the laboratory exercise, make sure to shutdown the system properly

**DONT‘S**

1. Students bringing the bags inside the laboratory..
2. Students wearing slippers/shoes insides the laboratory.
3. Students using the computers in an improper way.
4. Students scribbling on the desk and mishandling the chairs.
5. Students using mobile phones inside the laboratory.
6. Students making noise inside the laboratory.

**EXPERIMENT / PRACTICAL -1**

**Write a C program that contains a string (char pointer) with a value ‘Hello world’. The program should XOR each character in this string with 0 and displays the result.**

**OBJECTIVE OF THE EXPERIMENT:**The objective of the program is to perform an XOR operation on each character of the string "Hello World" with 0**.**

**OUTCOME OF THE EXPERIMENT:**Performing an XOR operation on each character of the string “Hello World” with 0 results the same string “Hello World” **.** The XOR operation outputs true if the inputs are not alike; otherwise, it outputs false.  
Here’s a step-by-step explanation of the outcomes of C program:  
 1. **Input:** Input for the program has already been declared with "Hello World". Instead of   
 prompting the user.

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **R** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

2. **XOR Operation:**

It performs logical operation. If input bits are the same, then the output will be false(0)   
 elsetrue(1).  
  
**3. Output:** The program outputs the same string “Hello World”. Because when we perform XOR operation with each character of the given string with 0 it produces the same character given as input. We know that performing XOR with 0 results the output same as given input.

**SYSTEM REQUIREMENTS:**

To run a C program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **C :**  Make sure you have C installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your C code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** C is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** C is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running C programs.

Once your system meets these requirements, you can create a C file (e.g., **my\_program.c**) and write the C program to compute the objective of experiment, as shown in the procedure. Save the file and run it using the C compiler in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.c file is located and run the following command:  
  
gcc my\_program.c  
  
If everything is set up correctly, the program will execute, and you should see the output.  
  
**ALGORITHM / PROCEDURE:**

1. Take the input string from the user.
2. Create a pointer pointing to the input string.
3. Loop through the pointer until it reaches to the end of input string (i.e '\0')
4. For each iteration perform XOR operation with 0
5. Finally, print the input string.

**PSEUDOCODE:**

#include <stdio.h>

void main() {

char text[] = "Hello world";

printf("Input: %s \n", text);

for (char \*ptr = text; \*ptr != '\0'; ptr++)

\*ptr = \*ptr ^ 0;

// Ptr is a pointer that will give us each char

// of 'text' one by one

// \0 represents end of string in C, thus we

// are using it in our condition

// ptr++ will make the pointer point to the next char

// in the text string

printf("After XOR: %s \n", text);

}

**EXPECTED OUTPUT:**

**Input: Hello world**

**After XOR: Hello world**

**RESULTS:**

**Note:** Include print copy of System generated actual output

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**PRE LAB QUESTIONS**Here are some pre-lab questions for the C program that computes the objective of the experiment:

1. How does the XOR operation work at the binary level?

2. What is the truth table for XOR?

3. What is the XOR operation, and what does it do?

4. How do you initialize a char pointer with the value "Hello world"?

5. What does XORing a character with 0 achieve?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voce questions for the C program that computes the objective of the experiment :  
 1.What is the purpose of the program?

2. How is the XOR operation performed on characters?

3. How is the result of each XOR operation displayed?

4. How would the output change if the XOR operation was performed with a value other than 0?

**EXPERIMENT / PRACTICAL -2**

**Writea C program that contains a string (char pointer) with a value ‘Hello world’. The program should AND or and XOR each character in this string with 127 and display the result.**

**OBJECTIVE OF THE EXPERIMENT:**The objective of the program is to perform AND, OR and XOR operations on each characters of the string “Hello World” with 127 and display the result.

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **R** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**OUTCOME OF THE EXPERIMENT:**The outcome of the program to perform AND, OR and XOR operation on each character of the string “Hello World” will results producing different string.  
Here’s a step-by-step explanation of the outcomes of C program:  
 **1. Input:** Input for the program has already been declared with "Hello World". Instead of  
 prompting the user.  
 2. **AND operation:**

Performing a bitwise AND operation between the ASCII value of the letter and the   
 value 127 involves comparing and combining the individual bits of the two values  
 according to the rules of the bitwise AND operation.

**For example let us try performing E^127**

Asci value of E is 69.

Convert 69 (decimal) to binary: 69= 64 + 4 + 1+2^6 + 2^2 + 2^0=1000101(binary)

Convert 127 (decimal) to binary: 127 = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 2^6 + 2^5 + 2^4

+ 2^3 + 2^2 + 2^1 + 2^0 = 1111111 (binary)   
 Perform the AND operation bit by bit:

1000101

&1111111

--------------

1000101

Convert the binary result back to decimal:

The binary value 1000101 represents the decimal number 69. In the ASCII character set, the

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **R** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

decimal value 69 corresponds to the character 'E'.  
 **3. OR operation:**

Performing a bitwise OR operation between the ASCII value of the letter and the value 127 involves comparing and combining the individual bits of the two values according to the rules of the bitwise OR operation.

For example let us try performing E^127

Asci value of E is 69.

Convert 69 (decimal) to binary:

69=64 + 4 + 1+2^6 + 2^2 + 2^0=1000101(binary)

Convert 127 (decimal) to binary:  
 127 = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1111111 (binary)

Perform the XOR operation bit by bit:

1111111

**|** 1000101

-------------

1111111

Convert the binary result back to decimal:

1111111=127(decimal)

In the ASCII character set, the character with the decimal value 127 is often referred to as

the "delete" character.

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **R** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**4. XOR operation:**

Performing XOR operation on characters of string “Hello World” with 127 provides

different character.

**For example let us try performing A^127**

Ascii value of A is 65.

Convert 65 (decimal) to binary:

65 = 64 + 1 = 2^6 + 2^0 = 1000001 (binary)

Convert 127 (decimal) to binary:

127 = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1111111

(binary)

Perform the XOR operation bit by bit:

1000001 (ASCII 'A')

1111111 (Value 127)

-----------

0111110 (Result)

Convert the binary result back to decimal:

0111110 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 = 62 (decimal)

This value, when treated as a character in the ASCII character set, corresponds to the

character '>'

**5.Output :** The program will display the original characters, the results of the AND,  
 OR, and XOR operations, along with their ASCII values, for each character in the

string "Hello world" when combined with the value 127.

**SYSTEM REQUIREMENTS:**

To run a C program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **C :**  Make sure you have C installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your C code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** C is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** C is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running C programs.

Once your system meets these requirements, you can create a C file (e.g., **my\_program.c**) and write the C program to compute the objective of experiment, as shown in the procedure. Save the file and run it using the C compiler in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.c file is located and run the following command:  
  
gcc my\_program.c  
  
If everything is set up correctly, the program will execute, and you should see the output.  
  
**ALGORITHM / PROCEDURE:**

1. Create a total of three copies of the input string.
2. Create three pointers, each corresponding to the three strings.
3. Loop through one of the pointers until it reaches to the end of input string (i.e '\0')

    For each iteration:

* 1. Pointer #1 performs XOR with 127
  2. Pointer #2 performs OR with 127
  3. Pointer #3 performs AND with 127

1. Finally print the strings.

**PSEUDOCODE:**

#include <stdio.h>

#include <string.h>

void main() {

char text[] = "Hello world";

char \*text1 = strdup(text);

// Equivalent to char text1[] = "Hello world"

char \*text2 = strdup(text);

printf("Input: %s \n", text);

// Three pointers for three strings

char \*p1 = text, \*p2 = text1, \*p3 = text2;

// Since the three strings are of equal length

// we can just use p1 in the loop condition

while (\*p1 != '\0') {

\*p1 = \*p1 ^ 127;

\*p2 = \*p2 & 127;

\*p3 = \*p3 | 127;

p1++, p2++, p3++;

}

printf("XOR output: %s \n", text);

printf("AND output: %s \n", text1);

printf("OR output: %s \n", text2);

}

**OUTPUT:**

**Input: Hello world**

**XOR output: 7\_**

**AND output: Hello world**

**OR output:**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the C program that computes the objective of the experiment:

1.Do you understand how bitwise operations work at the binary level?

2.Do you know the ASCII representation of characters? How will this knowledge help

you in performing bitwise operations on the characters?

3.Can you explain what bitwise AND, OR, and XOR operations are in programming?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the C program that computes the objective of the experiment:

1. What is the significance of bitwise AND, OR, and XOR operations?

2. How did you perform the AND operation on each character of the string with 127?

3. Could you describe the steps you took to perform the OR operation on each

character with 127?

4. What is the difference between bitwise AND, OR, and XOR operations?

5. Did you encounter any challenges while implementing the program? How did you

overcome them?

**EXPERIMENT / PRACTICAL -3**

**Write a Java program to perform encryption and decryption using the following algorithms**

**a. Ceaser cipher**

**b. Substitution cipher**

**c. Hill Cipher**

**d. Play fair Cipher**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this Java program is to understand and implement encryption and decryption mechanisms using four classical ciphers: Ceaser Cipher, Substitution Cipher, Hill Cipher, and Play Fair Cipher. By exploring these ciphers, the experiment aims to familiarize students with the principles of basic cryptography, matrix operations, and string manipulations in Java.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion of this Java program:

The user will be able to encrypt a plaintext message using the Ceaser Cipher, Substitution Cipher, Hill Cipher, and Play Fair Cipher algorithms.

Similarly, the user will be able to decrypt an encrypted message back to its original plaintext format using the same algorithms.

Students will have a deeper understanding of the mechanisms and vulnerabilities of each cipher.

Participants will be able to appreciate the evolution of encryption techniques and their significance in data protection and security.

The students will gain hands-on experience in Java programming, focusing on its string manipulation capabilities, matrix operations, and data structures.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

## Caesar:

1. For each character in the input string, do:
   1. Convert the character to lowercase.
   2. Subtract 'a' from it, to convert ASCII to an effective value of 0 for us to work on.
   3. If the mode is "encrypt"

1.3.1 Add integer '3' to the integer 'c'

* 1. Else

1.4.1 Subtract the integer '3' from 'c'.

* 1. If the integer becomes negative

1.5.1 Add 26 to the integer (because modulo operation does not work on negative

integers in Java)

* 1. else perform modulo 26 on the integer.
  2. Finally convert the number back to a character by adding 'a' and typecasting it using (char)

## Substitution:

1. For each character in the input string, do:
   1. Convert the character to lowercase.
   2. Subtract 'a' from it, to convert ASCII to an effective value of 0 for us to work on.
   3. If the mode is "encrypt"

1.3.1 Add the key integer to the integer 'c'

* 1. Else
     1. Subtract the key integer.
  2. If the integer becomes negative

1.5.1 Add 26 to the integer (because modulo operation does not work on negative integers

in Java)

* 1. else perform modulo 26 on the integer.
  2. Finally convert the number back to a character by adding 'a' and typecasting it using (char)

## Hill Cipher:

1. **Key Generation:**

* Choose a matrix key, called the encryption key, of size n x n, where n is a positive integer.
* Ensure that the matrix is invertible (its determinant is non-zero) to enable decryption.
* The elements of the matrix should typically be chosen from the set of integers modulo a chosen prime number to ensure security.

1. **Text Preparation:**

* Convert the plaintext message into numbers using a suitable mapping (e.g., A=0, B=1, ..., Z=25).
* If the message length is not a multiple of n, pad it with filler characters to make it fit.

1. **Matrix Formation:**

* Break the numeric representation of the plaintext message into blocks of size n each.
* Convert each block of n numbers into a column vector.

1. **Encryption:**

* For each column vector representing a block of plaintext numbers, multiply it by the encryption key matrix.
* Perform the matrix multiplication modulo the chosen prime to keep the results within a certain range.
* The result of each multiplication is a new column vector.

1. **Ciphertext Formation:**

* Convert each resulting column vector back to a numeric block.
* Convert each numeric block back to characters using the reverse mapping from step 2.

1. **Ciphertext Output:**

* Concatenate the character blocks to obtain the encrypted ciphertext.

## Play Fair Cipher:

**Playfair Cipher Encryption Algorithm:**

1. **Key Generation:**

* Convert the keyword to uppercase and replace 'J' with 'I'.
* Initialize a 5x5 matrix called keyTable for the Playfair matrix.
* Create an array taken to track the usage of letters in the matrix.
* Iterate through the keyword and fill the keyTable with unique letters.
* Fill the remaining positions in keyTable with unused letters from the alphabet.

1. **Text Preparation:**

* Convert the input plaintext to uppercase and replace 'J' with 'I'.
* Remove any non-alphabet characters from the plaintext.

1. **Encryption:**

* Iterate through the plaintext in pairs (digraphs).
* For each digraph:
* Find the positions of the letters in the keyTable.
* If both letters are in the same row:
* Replace each letter with the letter to its right.
* If both letters are in the same column:
* Replace each letter with the letter below it.
* If the letters are in different rows and columns:
* Replace each letter with the letter in the same row but the column of the other letter.

1. **Ciphertext Formation:**

* Concatenate the modified digraphs to obtain the encrypted ciphertext.

**Playfair Cipher Decryption Algorithm:**

1. **Decryption:**

* Iterate through the ciphertext in pairs (digraphs).
* For each digraph:
* Find the positions of the letters in the keyTable.
* If both letters are in the same row:
* Replace each letter with the letter to its left.
* If both letters are in the same column:
* Replace each letter with the letter above it.
* If the letters are in different rows and columns:
* Replace each letter with the letter in the same row but the column of the other letter.

1. **Plaintext Formation:**

* Concatenate the modified digraphs to obtain the decrypted plaintext.

**PSEUDOCODE:**

// This program handles both Caesar and Substitution Ciphers.

class SubsCrypt {

static String caesarEncrypt(String message) {

return operate(message, 3, true);

};

static String substitutionEncrypt(String message, int key) {

return operate(message, key, true);

}

static String caesarDecrypt(String message) {

return operate(message, 3, false);

};

static String substitutionDecrypt(String message, int key) {

return operate(message, key, false);

}

static String operate(String message, int key, boolean encrypt) {

String cipher = new String();

for (int i=0; i<message.length(); i++) {

int c = message.charAt(i);

if (Character.isAlphabetic(c) == false) {

cipher += (char)c;

continue;

}

c = Character.toLowerCase(c) - 'a';

if (encrypt == true)

c = c + key;

else

c = c - key;

if (c < 0)

c = 26 + c;

else

c = c % 26;

cipher += (char)(c + 'a');

}

return cipher;

}

public static void main(String args[]) {

String testCases[] = {

"ABC", "xyz", "skran", "skRAN",

"Haytham Kenway"

};

int key = 23;

for (String i:testCases) {

System.out.println("Message: "+ i);

String cipherText = SubsCrypt.caesarEncrypt(i);

System.out.println("CipherText (Caesar): "+ cipherText);

System.out.println("Decryption: "+ SubsCrypt.caesarDecrypt(cipherText));

cipherText = SubsCrypt.substitutionEncrypt(i, key);

System.out.printf("CipherText(Substitution) with key %d: %s", key, cipherText);

System.out.println("Decryption: "+ SubsCrypt.substitutionDecrypt(cipherText, key));

System.out.println();

}

}

}

**OUTPUT:**

**Message: Haytham Kenway**

**CipherText (Caesar): kdbwkdp nhqzdb**

**Decryption: haytham kenway**

**CipherText(Substitution) with key 23: exvqexj hbktxv**

**Decryption: haytham kenway**

**//program for Hill Cipher**

import java.util.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class HillCipherExample {

int[] lm;

int[][] km;

int[] rm;

static int choice;

int [][] invK;

public void performDivision(String temp, int s)

{

while (temp.length() > s)

{

String line = temp.substring(0, s);

temp = temp.substring(s, temp.length());

calLineMatrix(line);

if(choice ==1){

multiplyLineByKey(line.length());

}else{

multiplyLineByInvKey(line.length());

}

showResult(line.length());

}

if (temp.length() == s){

if(choice ==1){

calLineMatrix(temp);

multiplyLineByKey(temp.length());

showResult(temp.length());

}

else{

calLineMatrix(temp);

this.multiplyLineByInvKey(temp.length());

showResult(temp.length());

}

}

else if (temp.length() < s)

{

for (int i = temp.length(); i < s; i++)

temp = temp + 'x';

if(choice ==1){

calLineMatrix(temp);

multiplyLineByKey(temp.length());

showResult(temp.length());

}

else{

calLineMatrix(temp);

multiplyLineByInvKey(temp.length());

showResult(temp.length());

}

}

}

public void calKeyMatrix(String key, int len)

{

km = new int[len][len];

int k = 0;

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

km[i][j] = ((int) key.charAt(k)) - 97;

k++;

}

}

}

public void calLineMatrix(String line)

{

lm = new int[line.length()];

for (int i = 0; i < line.length(); i++)

{

lm[i] = ((int) line.charAt(i)) - 97;

}

}

public void multiplyLineByKey(int len)

{

rm = new int[len];

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

rm[i] += km[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public void multiplyLineByInvKey(int len)

{

rm = new int[len];

for (int i = 0; i < len; i++)

{

for (int j = 0; j < len; j++)

{

rm[i] += invK[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public void showResult(int len)

{

String result = "";

for (int i = 0; i < len; i++)

{

result += (char) (rm[i] + 97);

}

System.out.print(result);

}

public int calDeterminant(int A[][], int N)

{

int resultOfDet;

switch (N) {

case 1:

resultOfDet = A[0][0];

break;

case 2:

resultOfDet = A[0][0] \* A[1][1] - A[1][0] \* A[0][1];

break;

default:

resultOfDet = 0;

for (int j1 = 0; j1 < N; j1++)

{

int m[][] = new int[N - 1][N - 1];

for (int i = 1; i < N; i++)

{

int j2 = 0;

for (int j = 0; j < N; j++)

{

if (j == j1)

continue;

m[i - 1][j2] = A[i][j];

j2++;

}

}

resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) \* A[0][j1]

\* calDeterminant(m, N - 1);

} break;

}

return resultOfDet;

}

public void cofact(int num[][], int f)

{

int b[][], fac[][];

b = new int[f][f];

fac = new int[f][f];

int p, q, m, n, i, j;

for (q = 0; q < f; q++)

{

for (p = 0; p < f; p++)

{

m = 0;

n = 0;

for (i = 0; i < f; i++)

{

for (j = 0; j < f; j++)

{

b[i][j] = 0;

if (i != q && j != p)

{

b[m][n] = num[i][j];

if (n < (f - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

fac[q][p] = (int) Math.pow(-1, q + p) \* calDeterminant(b, f - 1);

}

}

trans(fac, f);

}

void trans(int fac[][], int r)

{

int i, j;

int b[][], inv[][];

b = new int[r][r];

inv = new int[r][r];

int d = calDeterminant(km, r);

int mi = mi(d % 26);

mi %= 26;

if (mi < 0)

mi += 26;

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

b[i][j] = fac[j][i];

}

}

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

inv[i][j] = b[i][j] % 26;

if (inv[i][j] < 0)

inv[i][j] += 26;

inv[i][j] \*= mi;

inv[i][j] %= 26;

}

}

//System.out.println("\nInverse key:");

//matrixtoinvkey(inv, r);

invK = inv;

}

public int mi(int d)

{

int q, r1, r2, r, t1, t2, t;

r1 = 26;

r2 = d;

t1 = 0;

t2 = 1;

while (r1 != 1 && r2 != 0)

{

q = r1 / r2;

r = r1 % r2;

t = t1 - (t2 \* q);

r1 = r2;

r2 = r;

t1 = t2;

t2 = t;

}

return (t1 + t2);

}

public void matrixtoinvkey(int inv[][], int n)

{

String invkey = "";

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

invkey += (char) (inv[i][j] + 97);

}

}

System.out.print(invkey);

}

public boolean check(String key, int len)

{

calKeyMatrix(key, len);

int d = calDeterminant(km, len);

d = d % 26;

if (d == 0)

{

System.out.println("Key is not invertible");

return false;

}

else if (d % 2 == 0 || d % 13 == 0)

{

System.out.println("Key is not invertible");

return false;

}

else

{

return true;

}

}

public static void main(String args[]) throws IOException

{

HillCipherExample obj = new HillCipherExample();

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Menu:\n1: Encryption\n2: Decryption");

choice = Integer.parseInt(in.readLine());

System.out.println("Enter the line: ");

String line = in.readLine();

System.out.println("Enter the key: ");

String key = in.readLine();

double sq = Math.sqrt(key.length());

if (sq != (long) sq)

System.out.println("Cannot Form a square matrix");

else

{

int size = (int) sq;

if (obj.check(key, size))

{

System.out.println("Result:");

obj.cofact(obj.km, size);

obj.performDivision(line, size);

}

}

}

}

**OUTPUT:**

**Menu:**

**1: Encryption**

**2: Decryption**

**1**

**Enter the line:**

**act**

**Enter the key:**

**gybnqkurp**

**Result:**

**poh**

**Menu:**

**1: Encryption**

**2: Decryption**

**2**

**Enter the line:**

**poh**

**Enter the key:**

**gybnqkurp**

**Result:**

**act**

**//program for Play Fair**

public class Play\_Fair {

public static void main(String[] args) {

String keyword = "Monarchy";

char[][] keyTable = generateKeyTable(keyword);

System.out.println("Message: " + keyword);

String plaintext = formatInput("instruments");

String ciphertext = encrypt(plaintext, keyTable);

System.out.println("Encrypted: " + ciphertext);

String decryptedText = decrypt(ciphertext, keyTable);

System.out.println("Decrypted: " + decryptedText);

}

private static final String ALPHABET = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

private static char[][] generateKeyTable(String keyword) {

keyword = keyword.toUpperCase().replaceAll("[J]", "I");

boolean[] taken = new boolean[26];

char[][] keyTable = new char[5][5];

int row = 0, col = 0;

for (int i = 0; i < keyword.length(); i++) {

char c = keyword.charAt(i);

if (!taken[c - 'A']) {

keyTable[row][col] = c;

taken[c - 'A'] = true;

col++;

if (col == 5) {

col = 0;

row++;

}

}

}

for (int i = 0; i < ALPHABET.length(); i++) {

char c = ALPHABET.charAt(i);

if (c != 'J' && !taken[c - 'A']) {

keyTable[row][col] = c;

taken[c - 'A'] = true;

col++;

if (col == 5) {

col = 0;

row++;

}

}

}

return keyTable;

}

private static String formatInput(String input) {

input = input.toUpperCase().replaceAll("[J]", "I");

StringBuilder formatted = new StringBuilder();

for (int i = 0; i < input.length(); i++) {

char c = input.charAt(i);

if (c >= 'A' && c <= 'Z') {

formatted.append(c);

}

}

return formatted.toString();

}

private static String encrypt(String plaintext, char[][] keyTable) {

StringBuilder ciphertext = new StringBuilder();

for (int i = 0; i < plaintext.length(); i += 2) {

char a = plaintext.charAt(i);

char b = (i + 1 < plaintext.length()) ? plaintext.charAt(i + 1) : 'X';

int[] posA = findPosition(a, keyTable);

int[] posB = findPosition(b, keyTable);

if (posA[0] == posB[0]) {

ciphertext.append(keyTable[posA[0]][(posA[1] + 1) % 5]);

ciphertext.append(keyTable[posB[0]][(posB[1] + 1) % 5]);

} else if (posA[1] == posB[1]) {

ciphertext.append(keyTable[(posA[0] + 1) % 5][posA[1]]);

ciphertext.append(keyTable[(posB[0] + 1) % 5][posB[1]]);

} else {

ciphertext.append(keyTable[posA[0]][posB[1]]);

ciphertext.append(keyTable[posB[0]][posA[1]]);

}

}

return ciphertext.toString();

}

private static String decrypt(String ciphertext, char[][] keyTable) {

StringBuilder plaintext = new StringBuilder();

for (int i = 0; i < ciphertext.length(); i += 2) {

char a = ciphertext.charAt(i);

char b = (i + 1 < ciphertext.length()) ? ciphertext.charAt(i + 1) : 'X';

int[] posA = findPosition(a, keyTable);

int[] posB = findPosition(b, keyTable);

if (posA[0] == posB[0]) {

plaintext.append(keyTable[posA[0]][(posA[1] + 4) % 5]);

plaintext.append(keyTable[posB[0]][(posB[1] + 4) % 5]);

} else if (posA[1] == posB[1]) {

plaintext.append(keyTable[(posA[0] + 4) % 5][posA[1]]);

plaintext.append(keyTable[(posB[0] + 4) % 5][posB[1]]);

} else {

plaintext.append(keyTable[posA[0]][posB[1]]);

plaintext.append(keyTable[posB[0]][posA[1]]);

}

}

return plaintext.toString();

}

private static int[] findPosition(char c, char[][] keyTable) {

int[] position = new int[2];

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

if (keyTable[i][j] == c) {

position[0] = i;

position[1] = j;

return position;

}

}

}

return position;

}

}

**OUTPUT:**

**PlainText: HIDETHEGOLDINTHETREESTUMP**

**Key: playfairexample**

**Encrypted: BMODZBXDNABEKUDMUIXXKZZRYI**

**Decrypted: HIDETHEGOLDINTHETREESTUMPX**

**RESULTS:**

**Note:** Include print copy of System generated actual output

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the Java program that computes the objective of the experiment:

1. What are the primary differences between symmetric and asymmetric encryption techniques?
2. Explain the basic principle behind the Ceaser Cipher and how it provides security?
3. How does the Substitution Cipher differ from the Ceaser Cipher in terms of encryption methodology?
4. What is the significance of matrices in the Hill Cipher?
5. Describe the mechanism of the Play Fair Cipher and how it operates on digraphs**.**

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the Java program that computes the objective of the experiment:

1. How do you initialize the key matrix for the Play Fair Cipher in Java?
2. In the Hill Cipher, what will happen if the key matrix is not invertible?
3. Can the Ceaser Cipher be considered a special case of the Substitution Cipher? Explain why or why not.
4. How would you improve the security of the Ceaser Cipher without changing its basic principle?
5. Describe a potential vulnerability of the Substitution Cipher and suggest a way to mitigate it.
6. Why is it essential to ensure that the key matrix in Hill Cipher is invertible?
7. What are the main challenges in decrypting a message encrypted using the Play Fair Cipher without knowing the key?
8. How does the size of the key affect the security of an encryption algorithm?
9. In what scenarios would you recommend using these classical ciphers in modern times?
10. What modern encryption techniques have evolved from these classical ciphers?

**EXPERIMENT / PRACTICAL -4**

**Write a C/JAVA program to implement the DES algorithm logic.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this C/Java program is to implement the Data Encryption Standard (DES) algorithm, which has historically been a predominant symmetric-key encryption technique. By doing this, students will gain insights into block cipher encryption mechanisms, the importance of key generation, permutation, substitution, and the overall intricacies of ensuring secure data transmission.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion of this C/Java program:

Users will have a clear understanding of how the DES algorithm works, including its processes like initial permutation, key generation, Feistel structure, and final permutation.

Students will be able to encrypt plaintext messages using DES and subsequently decrypt ciphertext messages back into their original plaintext forms.

The intricacies of bitwise operations, including XOR, permutations, and substitutions, will be familiar to the participants.

Learners will recognize the historical significance of DES, its strengths, and its vulnerabilities, especially in the context of modern encryption standards.

Students will have enhanced their programming skills in C/Java, with a special focus on encryption logic, bit manipulations, and algorithmic implementation.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

1. **Key Generation:**

* Start with a 64-bit secret key.
* Apply the Permutation Choice 1 (PC-1) to the key, reducing it to 56 bits.
* Split the 56-bit key into two 28-bit halves, left and right.

1. **Key Schedule:**

* Generate 16 subkeys, one for each round.
* For each round, rotate the two 28-bit halves separately to the left by a certain number of bits (determined by the key schedule) to create a shifted key.
* Apply Permutation Choice 2 (PC-2) to each shifted key, resulting in a 48-bit subkey.

1. **Initial Permutation:**

* Apply the Initial Permutation (IP) to the 64-bit plaintext message.

1. **16 Rounds of Encryption:**

* For each round, perform the following steps:
* Expand the right half of the data from 32 bits to 48 bits using Expansion Permutation.
* XOR the expanded data with the 48-bit subkey for the round.
* Divide the result into eight 6-bit blocks and pass them through the S-boxes (substitution boxes) to obtain 32 bits of data.
* Perform a permutation on the output of the S-boxes using the Permutation (P) function.
* XOR the output of the permutation with the left half of the data from the previous round.
* Swap the left and right halves of the data.

1. **Final Permutation:**

* After 16 rounds, swap the left and right halves of the data one last time.
* Apply the Final Permutation (IP^-1) to obtain the final encrypted 64-bit ciphertext.

1. **Decryption:**

* Decrypting a ciphertext involves reversing the process:
* Use the subkeys in reverse order.
* Perform the same steps as encryption, but apply the subkeys in reverse and adjust the rounds accordingly.

**PSEUDOCODE:**

import java.math.BigInteger;

public class DES {

public static String xor(String a, String b) {

StringBuilder result = new StringBuilder();

for (int i = 0; i < a.length(); i++) {

if (a.charAt(i) == b.charAt(i)) {

result.append("0");

} else {

result.append("1");

}

}

return result.toString();

}

public static String hexToBinary(String hex) {

String binary = new BigInteger(hex, 16).toString(2);

int remainder = binary.length() % 4;

if (remainder > 0) {

int zerosToAdd = 4 - remainder;

String leadingZeros = "0".repeat(zerosToAdd);

binary = leadingZeros + binary;

}

return binary;

}

public static String binaryToHex(String binary) {

return new BigInteger(binary, 2).toString(16);

}

public static String permute(String input, int[] table, int outputLength) {

StringBuilder result = new StringBuilder();

for (int i=0; i<outputLength; i++) {

result.append(input.charAt(table[i] - 1));

}

return result.toString().substring(0, outputLength);

}

public static int binToDec(String binary) {

int ans = Integer.parseInt(binary, 2);

return Integer.parseInt(binary, 2);

}

public static String decToBin(int decimal) {

String binary = Integer.toBinaryString(decimal);

int zerosToAdd = 4 - binary.length();

if (zerosToAdd > 0) {

String leadingZeros = "0".repeat(zerosToAdd);

binary = leadingZeros + binary;

}

return binary;

}

public static String shiftLeft(String input, int shifts) {

return input.substring(shifts) + input.substring(0, shifts);

}

public static String encrypt(String pt, String[] rkb, String[] rk) {

String binaryPt = hexToBinary(pt);

// Initial Permutation

String initialPermuted = permute(binaryPt, initialPerm, 64);

//System.out.println("After initial permutation: " + binaryToHex(initialPermuted));

// Splitting

String left = initialPermuted.substring(0, 32);

String right = initialPermuted.substring(32, 64);

for (int i = 0; i < 16; i++) {

// Expansion D-box

String rightExpanded = permute(right, expD, 48);

// XOR RoundKey[i] and right\_expanded

String xorX = xor(rightExpanded, rkb[i]);

// S-boxes

StringBuilder sboxStr = new StringBuilder();

for (int j = 0; j < 8; j++) {

int row = binToDec("" + xorX.charAt(j \* 6) + xorX.charAt(j \* 6 + 5));

int col = binToDec(""+xorX.charAt(j\*6 + 1) + xorX.charAt(j\*6 + 2) + xorX.charAt(j\*6 + 3) + xorX.charAt(j\*6 + 4));

int val = sbox[j][row][col];

sboxStr.append(decToBin(val));

}

// Straight D-box

String sboxPermuted = permute(sboxStr.toString(), per, 32);

// XOR left and sbox\_str

String result = xor(left, sboxPermuted);

left = result;

// Swapper

if (i != 15) {

String temp = left;

left = right;

right = temp;

}

//System.out.println("Round " + (i + 1) + " " + binaryToHex(left) + " " + binaryToHex(right) + " " + rk[i]);

}

// Combination

String combined = left + right;

// Final permutation

String cipherText = permute(combined, finalPerm, 64);

return cipherText;

}

public static void main(String[] args) {

String pt = "123456ABCD132536";

String key = "AABB09182736CCDD";

// Key generation

key = hexToBinary(key);

int[] keyp = {57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18, 10, 2,

59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36, 63, 55, 47, 39,

31, 23, 15, 7, 62, 54, 46, 38, 30, 22, 14, 6, 61, 53, 45, 37,

29, 21, 13, 5, 28, 20, 12, 4};

// Getting 56 bit key from 64 bit using the parity bits

key = permute(key, keyp, 56);

int[] shiftTable = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

int[] keyComp = {14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10, 23, 19, 12, 4,

26, 8, 16, 7, 27, 20, 13, 2, 41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32};

String left = key.substring(0, 28);

String right = key.substring(28, 56);

String[] rkb = new String[16];

String[] rk = new String[16];

for (int i = 0; i < 16; i++) {

left = shiftLeft(left, shiftTable[i]);

right = shiftLeft(right, shiftTable[i]);

String combineStr = left + right;

String roundKey = permute(combineStr, keyComp, 48);

rkb[i] = roundKey;

rk[i] = binaryToHex(roundKey);

}

System.out.println("Plain Text: " + pt);

String cipherText = encrypt(pt, rkb, rk);

cipherText = binaryToHex(cipherText);

System.out.println("Cipher Text: " + cipherText);

String[] rkbRev = new String[16];

String[] rkRev = new String[16];

for (int i = 0; i < 16; i++) {

rkbRev[i] = rkb[15 - i];

rkRev[i] = rk[15 - i];

}

String text = encrypt(cipherText, rkbRev, rkRev);

text = binaryToHex(text);

System.out.println("Plain Text: " + text);

}

static final int[] initialPerm = {

58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7 };

static final int[] expD = {

32, 1, 2, 3, 4, 5, 4, 5,

6, 7, 8, 9, 8, 9, 10, 11,

12, 13, 12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21, 20, 21,

22, 23, 24, 25, 24, 25, 26, 27,

28, 29, 28, 29, 30, 31, 32, 1

};

static final int[] per = {

16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26,

5, 18, 31, 10, 2, 8, 24, 14, 32, 27, 3, 9,

19, 13, 30, 6, 22, 11, 4, 25

};

static final int[][][] sbox = {

{{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7},

{0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8},

{4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0},

{15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13}},

{{15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10},

{3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5},

{0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15},

{13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9}},

{{10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8},

{13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1},

{13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7},

{1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12}},

{{7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15},

{13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9},

{10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4},

{3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14}},

{{2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9},

{14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6},

{4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14},

{11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3}},

{{12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11},

{10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8},

{9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6},

{4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13}},

{{4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1},

{13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6},

{1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2},

{6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12}},

{{13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7},

{1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2},

{7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8},

{2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11}}

};

static final int[] finalPerm = {

40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25 };

}

**PSEUDOCODE:**

**PlainText: 123456ABCD132536**

**Key: AABB09182736CCDD**

**CipherText: c0b7a8d05f3a829c**

**Decrypted: 123456abcd132536**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. What is the block size of the DES algorithm, and how does it affect the encryption process?
2. Describe the basic structure of the Feistel network used in DES.
3. How many rounds does DES use in its encryption and decryption processes?
4. What is the role of the S-boxes (Substitution boxes) in DES?
5. Explain the significance of key expansion and key scheduling in DES.

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. Why did the need for transitioning from DES to Triple DES (3DES) and eventually to AES arise?
2. How does the DES algorithm ensure that decryption is possible, given its multiple rounds of encryption?
3. In DES, what is the purpose of the initial and final permutations? Are they crucial for encryption security?
4. How do differential and linear cryptanalysis pose threats to DES?
5. If you were to compare DES with modern encryption algorithms, what would be its major shortcomings?
6. How does the "bit selection table" function in the DES algorithm?
7. What's the difference between a block cipher like DES and a stream cipher in terms of encryption?
8. Explain the parity bits in the 64-bit DES key and their purpose.
9. How would you mitigate the vulnerabilities of DES if you had to use it in today's world?
10. Can DES be considered secure for contemporary encryption needs? Why or why not?

**EXPERIMENT / PRACTICAL -5**

**Write a C/JAVA program to implement the Blowfish algorithm logic**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this C/Java program is to implement the Blowfish encryption algorithm, an advanced symmetric block cipher known for its simplicity, speed, and security. By exploring the Blowfish algorithm, students will delve into the nuances of modern encryption mechanisms, the design of S-boxes, and the importance of variable-length key structures.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion of this C/Java program:

Users will have a comprehensive understanding of the Blowfish algorithm, including its architecture, key expansion, and the Feistel network operations.

Students will be capable of encrypting plaintext messages using Blowfish and, in turn, decrypting the corresponding ciphertext messages.

The participants will develop a deep understanding of the bitwise operations, including XOR, and the importance of dynamic S-boxes which are key-dependent in Blowfish.

Learners will appreciate the adaptability of Blowfish in applications where the key does not change often, such as file encryption.

With a hands-on implementation, students will improve their programming prowess in C/Java, particularly regarding encryption logic, bit manipulations, and algorithmic flow.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

1. **Key Setup:**

* Initialize a P-array (18 32-bit subkeys) and four S-boxes (each with 256 32-bit entries).
* Expand the user-provided key into the P-array and S-boxes using a key expansion algorithm.

1. **Initial Permutation:**

* Divide the 64-bit block of plaintext into two 32-bit halves: L (left) and R (right).
* Apply an Initial Permutation (IP) to both L and R.

1. **16 Rounds of Encryption:**

* For each round (from 1 to 16), perform the following steps:
* XOR L with a subkey from the P-array.
* Pass L through the S-boxes, substituting 4 bytes of L with 4 bytes from the S-boxes.
* Perform a bitwise XOR between the output of the S-boxes and R.
* Swap L and R (swap L with the new R).
* After 16 rounds, swap L and R one final time.

1. **Final Permutation:**

* Apply a Final Permutation (IP^-1) to the final L and R to obtain the encrypted 64-bit ciphertext block.

1. **Decryption:**

* Decrypting a ciphertext block involves reversing the encryption process:
* Apply the key setup to generate the subkeys.
* Apply the initial permutation to the ciphertext block.
* Perform the rounds in reverse order, using the same subkeys.

**PSEUDOCODE:**

// Java Program to demonstrate Blowfish encryption

import java.util.Arrays;

public class Blowfish {

// Substitution boxes each string is a 32 bit hexadecimal value.

final String S[][]

= { { "d1310ba6", "98dfb5ac", "2ffd72db", "d01adfb7", "b8e1afed",

"6a267e96", "ba7c9045", "f12c7f99", "24a19947", "b3916cf7",

"0801f2e2", "858efc16", "636920d8", "71574e69", "a458fea3",

"f4933d7e", "0d95748f", "728eb658", "718bcd58", "82154aee",

"7b54a41d", "c25a59b5", "9c30d539", "2af26013", "c5d1b023",

"286085f0", "ca417918", "b8db38ef", "8e79dcb0", "603a180e",

"6c9e0e8b", "b01e8a3e", "d71577c1", "bd314b27", "78af2fda",

"55605c60", "e65525f3", "aa55ab94", "57489862", "63e81440",

"55ca396a", "2aab10b6", "b4cc5c34", "1141e8ce", "a15486af",

"7c72e993", "b3ee1411", "636fbc2a", "2ba9c55d", "741831f6",

"ce5c3e16", "9b87931e", "afd6ba33", "6c24cf5c", "7a325381",

"28958677", "3b8f4898", "6b4bb9af", "c4bfe81b", "66282193",

"61d809cc", "fb21a991", "487cac60", "5dec8032", "ef845d5d",

"e98575b1", "dc262302", "eb651b88", "23893e81", "d396acc5",

"0f6d6ff3", "83f44239", "2e0b4482", "a4842004", "69c8f04a",

"9e1f9b5e", "21c66842", "f6e96c9a", "670c9c61", "abd388f0",

"6a51a0d2", "d8542f68", "960fa728", "ab5133a3", "6eef0b6c",

"137a3be4", "ba3bf050", "7efb2a98", "a1f1651d", "39af0176",

"66ca593e", "82430e88", "8cee8619", "456f9fb4", "7d84a5c3",

"3b8b5ebe", "e06f75d8", "85c12073", "401a449f", "56c16aa6",

"4ed3aa62", "363f7706", "1bfedf72", "429b023d", "37d0d724",

"d00a1248", "db0fead3", "49f1c09b", "075372c9", "80991b7b",

"25d479d8", "f6e8def7", "e3fe501a", "b6794c3b", "976ce0bd",

"04c006ba", "c1a94fb6", "409f60c4", "5e5c9ec2", "196a2463",

"68fb6faf", "3e6c53b5", "1339b2eb", "3b52ec6f", "6dfc511f",

"9b30952c", "cc814544", "af5ebd09", "bee3d004", "de334afd",

"660f2807", "192e4bb3", "c0cba857", "45c8740f", "d20b5f39",

"b9d3fbdb", "5579c0bd", "1a60320a", "d6a100c6", "402c7279",

"679f25fe", "fb1fa3cc", "8ea5e9f8", "db3222f8", "3c7516df",

"fd616b15", "2f501ec8", "ad0552ab", "323db5fa", "fd238760",

"53317b48", "3e00df82", "9e5c57bb", "ca6f8ca0", "1a87562e",

"df1769db", "d542a8f6", "287effc3", "ac6732c6", "8c4f5573",

"695b27b0", "bbca58c8", "e1ffa35d", "b8f011a0", "10fa3d98",

"fd2183b8", "4afcb56c", "2dd1d35b", "9a53e479", "b6f84565",

"d28e49bc", "4bfb9790", "e1ddf2da", "a4cb7e33", "62fb1341",

"cee4c6e8", "ef20cada", "36774c01", "d07e9efe", "2bf11fb4",

"95dbda4d", "ae909198", "eaad8e71", "6b93d5a0", "d08ed1d0",

"afc725e0", "8e3c5b2f", "8e7594b7", "8ff6e2fb", "f2122b64",

"8888b812", "900df01c", "4fad5ea0", "688fc31c", "d1cff191",

"b3a8c1ad", "2f2f2218", "be0e1777", "ea752dfe", "8b021fa1",

"e5a0cc0f", "b56f74e8", "18acf3d6", "ce89e299", "b4a84fe0",

"fd13e0b7", "7cc43b81", "d2ada8d9", "165fa266", "80957705",

"93cc7314", "211a1477", "e6ad2065", "77b5fa86", "c75442f5",

"fb9d35cf", "ebcdaf0c", "7b3e89a0", "d6411bd3", "ae1e7e49",

"00250e2d", "2071b35e", "226800bb", "57b8e0af", "2464369b",

"f009b91e", "5563911d", "59dfa6aa", "78c14389", "d95a537f",

"207d5ba2", "02e5b9c5", "83260376", "6295cfa9", "11c81968",

"4e734a41", "b3472dca", "7b14a94a", "1b510052", "9a532915",

"d60f573f", "bc9bc6e4", "2b60a476", "81e67400", "08ba6fb5",

"571be91f", "f296ec6b", "2a0dd915", "b6636521", "e7b9f9b6",

"ff34052e", "c5855664", "53b02d5d", "a99f8fa1", "08ba4799",

"6e85076a" },

{ "4b7a70e9", "b5b32944", "db75092e", "c4192623", "ad6ea6b0",

"49a7df7d", "9cee60b8", "8fedb266", "ecaa8c71", "699a17ff",

"5664526c", "c2b19ee1", "193602a5", "75094c29", "a0591340",

"e4183a3e", "3f54989a", "5b429d65", "6b8fe4d6", "99f73fd6",

"a1d29c07", "efe830f5", "4d2d38e6", "f0255dc1", "4cdd2086",

"8470eb26", "6382e9c6", "021ecc5e", "09686b3f", "3ebaefc9",

"3c971814", "6b6a70a1", "687f3584", "52a0e286", "b79c5305",

"aa500737", "3e07841c", "7fdeae5c", "8e7d44ec", "5716f2b8",

"b03ada37", "f0500c0d", "f01c1f04", "0200b3ff", "ae0cf51a",

"3cb574b2", "25837a58", "dc0921bd", "d19113f9", "7ca92ff6",

"94324773", "22f54701", "3ae5e581", "37c2dadc", "c8b57634",

"9af3dda7", "a9446146", "0fd0030e", "ecc8c73e", "a4751e41",

"e238cd99", "3bea0e2f", "3280bba1", "183eb331", "4e548b38",

"4f6db908", "6f420d03", "f60a04bf", "2cb81290", "24977c79",

"5679b072", "bcaf89af", "de9a771f", "d9930810", "b38bae12",

"dccf3f2e", "5512721f", "2e6b7124", "501adde6", "9f84cd87",

"7a584718", "7408da17", "bc9f9abc", "e94b7d8c", "ec7aec3a",

"db851dfa", "63094366", "c464c3d2", "ef1c1847", "3215d908",

"dd433b37", "24c2ba16", "12a14d43", "2a65c451", "50940002",

"133ae4dd", "71dff89e", "10314e55", "81ac77d6", "5f11199b",

"043556f1", "d7a3c76b", "3c11183b", "5924a509", "f28fe6ed",

"97f1fbfa", "9ebabf2c", "1e153c6e", "86e34570", "eae96fb1",

"860e5e0a", "5a3e2ab3", "771fe71c", "4e3d06fa", "2965dcb9",

"99e71d0f", "803e89d6", "5266c825", "2e4cc978", "9c10b36a",

"c6150eba", "94e2ea78", "a5fc3c53", "1e0a2df4", "f2f74ea7",

"361d2b3d", "1939260f", "19c27960", "5223a708", "f71312b6",

"ebadfe6e", "eac31f66", "e3bc4595", "a67bc883", "b17f37d1",

"018cff28", "c332ddef", "be6c5aa5", "65582185", "68ab9802",

"eecea50f", "db2f953b", "2aef7dad", "5b6e2f84", "1521b628",

"29076170", "ecdd4775", "619f1510", "13cca830", "eb61bd96",

"0334fe1e", "aa0363cf", "b5735c90", "4c70a239", "d59e9e0b",

"cbaade14", "eecc86bc", "60622ca7", "9cab5cab", "b2f3846e",

"648b1eaf", "19bdf0ca", "a02369b9", "655abb50", "40685a32",

"3c2ab4b3", "319ee9d5", "c021b8f7", "9b540b19", "875fa099",

"95f7997e", "623d7da8", "f837889a", "97e32d77", "11ed935f",

"16681281", "0e358829", "c7e61fd6", "96dedfa1", "7858ba99",

"57f584a5", "1b227263", "9b83c3ff", "1ac24696", "cdb30aeb",

"532e3054", "8fd948e4", "6dbc3128", "58ebf2ef", "34c6ffea",

"fe28ed61", "ee7c3c73", "5d4a14d9", "e864b7e3", "42105d14",

"203e13e0", "45eee2b6", "a3aaabea", "db6c4f15", "facb4fd0",

"c742f442", "ef6abbb5", "654f3b1d", "41cd2105", "d81e799e",

"86854dc7", "e44b476a", "3d816250", "cf62a1f2", "5b8d2646",

"fc8883a0", "c1c7b6a3", "7f1524c3", "69cb7492", "47848a0b",

"5692b285", "095bbf00", "ad19489d", "1462b174", "23820e00",

"58428d2a", "0c55f5ea", "1dadf43e", "233f7061", "3372f092",

"8d937e41", "d65fecf1", "6c223bdb", "7cde3759", "cbee7460",

"4085f2a7", "ce77326e", "a6078084", "19f8509e", "e8efd855",

"61d99735", "a969a7aa", "c50c06c2", "5a04abfc", "800bcadc",

"9e447a2e", "c3453484", "fdd56705", "0e1e9ec9", "db73dbd3",

"105588cd", "675fda79", "e3674340", "c5c43465", "713e38d8",

"3d28f89e", "f16dff20", "153e21e7", "8fb03d4a", "e6e39f2b",

"db83adf7" },

{ "e93d5a68", "948140f7", "f64c261c", "94692934", "411520f7",

"7602d4f7", "bcf46b2e", "d4a20068", "d4082471", "3320f46a",

"43b7d4b7", "500061af", "1e39f62e", "97244546", "14214f74",

"bf8b8840", "4d95fc1d", "96b591af", "70f4ddd3", "66a02f45",

"bfbc09ec", "03bd9785", "7fac6dd0", "31cb8504", "96eb27b3",

"55fd3941", "da2547e6", "abca0a9a", "28507825", "530429f4",

"0a2c86da", "e9b66dfb", "68dc1462", "d7486900", "680ec0a4",

"27a18dee", "4f3ffea2", "e887ad8c", "b58ce006", "7af4d6b6",

"aace1e7c", "d3375fec", "ce78a399", "406b2a42", "20fe9e35",

"d9f385b9", "ee39d7ab", "3b124e8b", "1dc9faf7", "4b6d1856",

"26a36631", "eae397b2", "3a6efa74", "dd5b4332", "6841e7f7",

"ca7820fb", "fb0af54e", "d8feb397", "454056ac", "ba489527",

"55533a3a", "20838d87", "fe6ba9b7", "d096954b", "55a867bc",

"a1159a58", "cca92963", "99e1db33", "a62a4a56", "3f3125f9",

"5ef47e1c", "9029317c", "fdf8e802", "04272f70", "80bb155c",

"05282ce3", "95c11548", "e4c66d22", "48c1133f", "c70f86dc",

"07f9c9ee", "41041f0f", "404779a4", "5d886e17", "325f51eb",

"d59bc0d1", "f2bcc18f", "41113564", "257b7834", "602a9c60",

"dff8e8a3", "1f636c1b", "0e12b4c2", "02e1329e", "af664fd1",

"cad18115", "6b2395e0", "333e92e1", "3b240b62", "eebeb922",

"85b2a20e", "e6ba0d99", "de720c8c", "2da2f728", "d0127845",

"95b794fd", "647d0862", "e7ccf5f0", "5449a36f", "877d48fa",

"c39dfd27", "f33e8d1e", "0a476341", "992eff74", "3a6f6eab",

"f4f8fd37", "a812dc60", "a1ebddf8", "991be14c", "db6e6b0d",

"c67b5510", "6d672c37", "2765d43b", "dcd0e804", "f1290dc7",

"cc00ffa3", "b5390f92", "690fed0b", "667b9ffb", "cedb7d9c",

"a091cf0b", "d9155ea3", "bb132f88", "515bad24", "7b9479bf",

"763bd6eb", "37392eb3", "cc115979", "8026e297", "f42e312d",

"6842ada7", "c66a2b3b", "12754ccc", "782ef11c", "6a124237",

"b79251e7", "06a1bbe6", "4bfb6350", "1a6b1018", "11caedfa",

"3d25bdd8", "e2e1c3c9", "44421659", "0a121386", "d90cec6e",

"d5abea2a", "64af674e", "da86a85f", "bebfe988", "64e4c3fe",

"9dbc8057", "f0f7c086", "60787bf8", "6003604d", "d1fd8346",

"f6381fb0", "7745ae04", "d736fccc", "83426b33", "f01eab71",

"b0804187", "3c005e5f", "77a057be", "bde8ae24", "55464299",

"bf582e61", "4e58f48f", "f2ddfda2", "f474ef38", "8789bdc2",

"5366f9c3", "c8b38e74", "b475f255", "46fcd9b9", "7aeb2661",

"8b1ddf84", "846a0e79", "915f95e2", "466e598e", "20b45770",

"8cd55591", "c902de4c", "b90bace1", "bb8205d0", "11a86248",

"7574a99e", "b77f19b6", "e0a9dc09", "662d09a1", "c4324633",

"e85a1f02", "09f0be8c", "4a99a025", "1d6efe10", "1ab93d1d",

"0ba5a4df", "a186f20f", "2868f169", "dcb7da83", "573906fe",

"a1e2ce9b", "4fcd7f52", "50115e01", "a70683fa", "a002b5c4",

"0de6d027", "9af88c27", "773f8641", "c3604c06", "61a806b5",

"f0177a28", "c0f586e0", "006058aa", "30dc7d62", "11e69ed7",

"2338ea63", "53c2dd94", "c2c21634", "bbcbee56", "90bcb6de",

"ebfc7da1", "ce591d76", "6f05e409", "4b7c0188", "39720a3d",

"7c927c24", "86e3725f", "724d9db9", "1ac15bb4", "d39eb8fc",

"ed545578", "08fca5b5", "d83d7cd3", "4dad0fc4", "1e50ef5e",

"b161e6f8", "a28514d9", "6c51133c", "6fd5c7e7", "56e14ec4",

"362abfce", "ddc6c837", "d79a3234", "92638212", "670efa8e",

"406000e0" },

{ "3a39ce37", "d3faf5cf", "abc27737", "5ac52d1b", "5cb0679e",

"4fa33742", "d3822740", "99bc9bbe", "d5118e9d", "bf0f7315",

"d62d1c7e", "c700c47b", "b78c1b6b", "21a19045", "b26eb1be",

"6a366eb4", "5748ab2f", "bc946e79", "c6a376d2", "6549c2c8",

"530ff8ee", "468dde7d", "d5730a1d", "4cd04dc6", "2939bbdb",

"a9ba4650", "ac9526e8", "be5ee304", "a1fad5f0", "6a2d519a",

"63ef8ce2", "9a86ee22", "c089c2b8", "43242ef6", "a51e03aa",

"9cf2d0a4", "83c061ba", "9be96a4d", "8fe51550", "ba645bd6",

"2826a2f9", "a73a3ae1", "4ba99586", "ef5562e9", "c72fefd3",

"f752f7da", "3f046f69", "77fa0a59", "80e4a915", "87b08601",

"9b09e6ad", "3b3ee593", "e990fd5a", "9e34d797", "2cf0b7d9",

"022b8b51", "96d5ac3a", "017da67d", "d1cf3ed6", "7c7d2d28",

"1f9f25cf", "adf2b89b", "5ad6b472", "5a88f54c", "e029ac71",

"e019a5e6", "47b0acfd", "ed93fa9b", "e8d3c48d", "283b57cc",

"f8d56629", "79132e28", "785f0191", "ed756055", "f7960e44",

"e3d35e8c", "15056dd4", "88f46dba", "03a16125", "0564f0bd",

"c3eb9e15", "3c9057a2", "97271aec", "a93a072a", "1b3f6d9b",

"1e6321f5", "f59c66fb", "26dcf319", "7533d928", "b155fdf5",

"03563482", "8aba3cbb", "28517711", "c20ad9f8", "abcc5167",

"ccad925f", "4de81751", "3830dc8e", "379d5862", "9320f991",

"ea7a90c2", "fb3e7bce", "5121ce64", "774fbe32", "a8b6e37e",

"c3293d46", "48de5369", "6413e680", "a2ae0810", "dd6db224",

"69852dfd", "09072166", "b39a460a", "6445c0dd", "586cdecf",

"1c20c8ae", "5bbef7dd", "1b588d40", "ccd2017f", "6bb4e3bb",

"dda26a7e", "3a59ff45", "3e350a44", "bcb4cdd5", "72eacea8",

"fa6484bb", "8d6612ae", "bf3c6f47", "d29be463", "542f5d9e",

"aec2771b", "f64e6370", "740e0d8d", "e75b1357", "f8721671",

"af537d5d", "4040cb08", "4eb4e2cc", "34d2466a", "0115af84",

"e1b00428", "95983a1d", "06b89fb4", "ce6ea048", "6f3f3b82",

"3520ab82", "011a1d4b", "277227f8", "611560b1", "e7933fdc",

"bb3a792b", "344525bd", "a08839e1", "51ce794b", "2f32c9b7",

"a01fbac9", "e01cc87e", "bcc7d1f6", "cf0111c3", "a1e8aac7",

"1a908749", "d44fbd9a", "d0dadecb", "d50ada38", "0339c32a",

"c6913667", "8df9317c", "e0b12b4f", "f79e59b7", "43f5bb3a",

"f2d519ff", "27d9459c", "bf97222c", "15e6fc2a", "0f91fc71",

"9b941525", "fae59361", "ceb69ceb", "c2a86459", "12baa8d1",

"b6c1075e", "e3056a0c", "10d25065", "cb03a442", "e0ec6e0e",

"1698db3b", "4c98a0be", "3278e964", "9f1f9532", "e0d392df",

"d3a0342b", "8971f21e", "1b0a7441", "4ba3348c", "c5be7120",

"c37632d8", "df359f8d", "9b992f2e", "e60b6f47", "0fe3f11d",

"e54cda54", "1edad891", "ce6279cf", "cd3e7e6f", "1618b166",

"fd2c1d05", "848fd2c5", "f6fb2299", "f523f357", "a6327623",

"93a83531", "56cccd02", "acf08162", "5a75ebb5", "6e163697",

"88d273cc", "de966292", "81b949d0", "4c50901b", "71c65614",

"e6c6c7bd", "327a140a", "45e1d006", "c3f27b9a", "c9aa53fd",

"62a80f00", "bb25bfe2", "35bdd2f6", "71126905", "b2040222",

"b6cbcf7c", "cd769c2b", "53113ec0", "1640e3d3", "38abbd60",

"2547adf0", "ba38209c", "f746ce76", "77afa1c5", "20756060",

"85cbfe4e", "8ae88dd8", "7aaaf9b0", "4cf9aa7e", "1948c25c",

"02fb8a8c", "01c36ae4", "d6ebe1f9", "90d4f869", "a65cdea0",

"3f09252d", "c208e69f", "b74e6132", "ce77e25b", "578fdfe3",

"3ac372e6" } };

// Subkeys initialisation with digits of pi.

final String P[] = { "243f6a88", "85a308d3", "13198a2e", "03707344", "a4093822",

"299f31d0", "082efa98", "ec4e6c89", "452821e6", "38d01377",

"be5466cf", "34e90c6c", "c0ac29b7", "c97c50dd", "3f84d5b5",

"b5470917", "9216d5d9", "8979fb1b" };

// storing 2^32 (for addition modulo 2^32).

// binary shifting 1 till it reaches the 32nd

// bit will turn modVal into our required

// value of 2^32

final long modVal = (1L << 32);

// to convert hexadecimal to binary.

private String hexToBin(String plainText)

{

String binary = "";

Long num;

String binary4B;

int n = plainText.length();

for (int i = 0; i < n; i++) {

num = Long.parseUnsignedLong(

plainText.charAt(i) + "", 16);

binary4B = Long.toBinaryString(num);

// each value in hexadecimal is 4 bits in binary.

binary4B = "0000" + binary4B;

binary4B = binary4B.substring(binary4B.length() - 4);

binary += binary4B;

}

return binary;

}

// convert from binary to hexadecimal.

private String binToHex(String plainText)

{

long num = Long.parseUnsignedLong(plainText, 2);

String hexa = Long.toHexString(num);

while (hexa.length() < (plainText.length() / 4))

// maintain output length same length

// as input by appending leading zeroes.

hexa = "0" + hexa;

return hexa;

}

// xor two hexadecimal strings of the same length.

private String xor(String a, String b)

{

a = hexToBin(a);

b = hexToBin(b);

String ans = "";

for (int i = 0; i < a.length(); i++)

ans += (char)(((a.charAt(i) - '0')

^ (b.charAt(i) - '0'))

+ '0');

ans = binToHex(ans);

return ans;

}

// addition modulo 2^32 of two hexadecimal strings.

private String addBin(String a, String b)

{

String ans = "";

long n1 = Long.parseUnsignedLong(a, 16);

long n2 = Long.parseUnsignedLong(b, 16);

n1 = (n1 + n2) % modVal;

ans = Long.toHexString(n1);

ans = "00000000" + ans;

return ans.substring(ans.length() - 8);

}

// function F explained above.

private String f(String plainText)

{

String a[] = new String[4];

String ans = "";

for (int i = 0; i < 8; i += 2) {

// the column number for S-box

// is 8-bit value(8\*4 = 32 bit plain text)

long col

= Long.parseUnsignedLong(

hexToBin(

plainText

.substring(i, i + 2)),

2);

a[i / 2] = S[i / 2][(int)col];

}

ans = addBin(a[0], a[1]);

ans = xor(ans, a[2]);

ans = addBin(ans, a[3]);

return ans;

}

// generate subkeys.

private String[] keyGenerate(String key)

{

int j = 0;

String subKeys[] = Arrays.copyOf(P, P.length);

for (int i = 0; i < subKeys.length; i++) {

// xor-ing 32-bit parts of the key

// with initial subkeys.

subKeys[i] = xor(subKeys[i], key.substring(j, j + 8));

// System.out.println("subkey "+ (i + 1) + ": "+ subKeys[i]);

j = (j + 8) % key.length();

}

return subKeys;

}

// round function

private String round(int time, String plainText, String[] subKeys)

{

String left, right;

left = plainText.substring(0, 8);

right = plainText.substring(8, 16);

left = xor(left, subKeys[time]);

// output from F function

String fOut = f(left);

right = xor(fOut, right);

// System.out.println("round " + time + ": " + right + left);

// swap left and right

return right + left;

}

// encryption

private String crypt(String text, String key, boolean encryptFlag)

{

String subKeys[] = keyGenerate(key);

if (encryptFlag == true) {

for (int i = 0; i < 16; i++)

text = round(i, text, subKeys);

} else {

for (int i = 17; i > 1; i--)

text = round(i, text, subKeys);

}

// postprocessing

String right = text.substring(0, 8);

String left = text.substring(8, 16);

if (encryptFlag == true) {

right = xor(right, subKeys[16]);

left = xor(left, subKeys[17]);

} else {

right = xor(right, subKeys[1]);

left = xor(left, subKeys[0]);

}

return left + right;

}

public static void main(String args[])

{

// Plain text must be Hex only (0-9 or a-f) for now

String plainText = "1234abcdef1234ab";

String key = "aabb09182736ccdd";

Blowfish bf = new Blowfish();

System.out.println("PlainText: " + plainText);

String cipherText = bf.crypt(plainText, key, true);

System.out.println("CipherText: " + cipherText);

String decrypted = bf.crypt(cipherText, key, false);

System.out.println("Decrypted: " + decrypted);

}

}

**OUTPUT:**

**PlainText: 1234abcdef1234ab**

**Key: aabb09182736ccdd**

**CipherText: 6ef29a33e55a05c1**

**Decrypted: 1234abcdef1234ab**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Who is the creator of the Blowfish algorithm, and what was the primary purpose behind its creation?
2. Explain the significance of the P-array and the S-boxes in the Blowfish algorithm.
3. How does the variable-length key feature in Blowfish enhance its security?
4. Describe the structure and operation of the Feistel network as employed by Blowfish.
5. What is the block size of the Blowfish algorithm, and how does it compare with other encryption algorithms?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. Why is Blowfish considered suitable for applications where the encryption key doesn't change frequently?
2. Discuss a potential vulnerability of the Blowfish algorithm when used with certain types of data patterns.
3. How does the key expansion phase in Blowfish contribute to its security, especially against brute force attacks?
4. In the context of Blowfish, why is it important that S-boxes are dynamically generated and key-dependent?
5. What are the primary distinctions between Blowfish and its successor, Twofish?
6. How does the Blowfish algorithm handle plaintexts that are not a multiple of its block size?
7. Describe the importance of the "splitting" process in Blowfish, where the 32-bit input is divided into two 16-bit halves.
8. Given the advancements in computational capabilities, is Blowfish still considered secure for modern encryption tasks?
9. How would you modify the Blowfish algorithm to make it more resistant to potential cryptographic attacks in the future?
10. Considering the cipher's design and its creator's intent, why was the Blowfish algorithm released to the public domain?

**EXPERIMENT / PRACTICAL -6**

**Write a C/JAVA program to implement the Rijndael algorithm logic.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this C/Java program is to implement the Rijndael encryption algorithm, which is the foundation of the Advanced Encryption Standard (AES). By diving into the workings of the Rijndael algorithm, students will learn the principles of symmetric block ciphers, the intricacies of state transformations, key expansions, and the vital role of S-boxes in ensuring data security.

**OUTCOME OF THE EXPERIMENT:**

Upon the successful execution of this C/Java program:

1. Users will be thoroughly acquainted with the architecture and functioning of the Rijndael algorithm, encompassing its various rounds, state transformations, and key expansion techniques.
2. Students will be proficient in encrypting plaintext messages using Rijndael and, subsequently, decrypting the encrypted messages back to plaintext.
3. Participants will gain insights into the operations of SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are pivotal in the Rijndael encryption process.
4. Learners will recognize the significance of the AES standard in today's cryptographic applications and understand how Rijndael was adapted for it.
5. By practically implementing the algorithm, students will bolster their programming expertise in C/Java, especially in the areas of encryption logic, matrix manipulations, and bitwise operations.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**Rijndael Algorithm (AES):**

1. **Key Setup:**

* Select a key size: AES supports key sizes of 128, 192, or 256 bits.
* Expand the key using the Rijndael key schedule to generate round keys.

1. **Initial Round:**

* Divide the input plaintext block into columns and rows forming a 4x4 matrix called the state.
* XOR each byte of the state with a corresponding byte of the round key.

1. **Rounds of Encryption (AES-128: 10 rounds, AES-192: 12 rounds, AES-256: 14 rounds):**

* Each round consists of the following steps:
* SubBytes: Substitute each byte of the state using a fixed substitution table (S-box).
* ShiftRows: Shift the rows of the state by varying numbers of bytes.
* MixColumns: Mix the columns of the state using a matrix multiplication.
* AddRoundKey: XOR each byte of the state with a round key.

1. **Final Round:**

* The final round omits the MixColumns step:
* SubBytes
* ShiftRows
* AddRoundKey

1. **Decryption:**

* The decryption process involves reversing the encryption steps using the same round keys in reverse order:
* InvShiftRows: Reverse the row shifts.
* InvSubBytes: Reverse the substitution using the inverse S-box.
* AddRoundKey: XOR each byte of the state with a round key.
* InvMixColumns: Reverse the MixColumns step (except in the final round).

**PSEUDOCODE:**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.\*;

public class Rijndael {

private static String asHex(byte buf[]) {

StringBuffer strbuf = new StringBuffer(buf.length\*2);

for (int i = 0; i < buf.length; i++) {

if(((int)buf[i]&0xff)<0x10)

strbuf.append("0");

strbuf.append(Long.toString((int)buf[i]&0xff,16));

}

return strbuf.toString();

}

public static void main(String[] args) throws Exception {

System.out.println("Enter message: ");

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String message = reader.readLine();

//Get the keyGenerator

KeyGenerator kgen = KeyGenerator.getInstance("AES");

kgen.init(128);

//Generate secret key specs

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

//Instantiate the cipher

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE, skeySpec);

byte[] encrypted = cipher.doFinal((args.length==0 ? message : args[0]).getBytes());

System.out.println("Encrypted String: " + asHex(encrypted));

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] original = cipher.doFinal(encrypted);

String originalString = new String(original);

System.out.println("Original String in Hexadecimal: " + asHex(original));

System.out.println("Original String: " + originalString);

}

}

**OUTPUT:**

**Rijndael/AES using built-in libraries**

**Enter message:**

**Namashkaar Mithro!**

**Encrypted String: 362db49dbd0f70cbc8c08676428f4b33ff3aa2a597f2955b69eed42a15b1b1d3**

**Decrypted (Hex form): 4e616d6173686b616172204d697468726f21**

**Decrypted: Namashkaar Mithro!**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Detail the history and reasons behind the selection of Rijndael as the basis for the AES standard.
2. What are the possible key and block sizes in the Rijndael algorithm, and how do they determine the number of rounds in the encryption process?
3. Describe the role and function of the SubBytes step in the Rijndael algorithm.
4. How do the ShiftRows and MixColumns operations contribute to the diffusion property in Rijndael?
5. Why is the key expansion phase crucial in the Rijndael encryption process?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. How does the structure of Rijndael's S-box contribute to its resistance against linear and differential cryptanalysis?
2. In what significant ways does AES differ from its precursor, Rijndael?
3. Explain the MixColumns transformation in the Rijndael algorithm using matrix multiplication in Galois Field.
4. How does Rijndael ensure decryption is possible after multiple rounds of complex encryption transformations?
5. Describe any potential vulnerabilities or criticisms faced by the Rijndael algorithm, and how they have been addressed.
6. How does the AddRoundKey operation in Rijndael work, and why is it considered an essential step for ensuring cryptographic strength?
7. What was the motivation behind developing a new encryption standard like AES, leading to the selection of Rijndael?
8. Can Rijndael be employed in a streaming mode? If so, how?
9. Why is AES (based on Rijndael) preferred in numerous security applications over older encryption methods like DES?
10. Describe the significance of the "counter mode" when using the Rijndael algorithm for specific cryptographic applications.

**EXPERIMENT / PRACTICAL -7**

**Write the RC4 logic in Java Using Java cryptography; encrypt the text “Hello world” using Blowfish. Create your own key using Java key tool.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this Java program is threefold:

1. Implement the RC4 encryption algorithm using Java Cryptography Extension (JCE).

2. Encrypt the text "Hello world" using the Blowfish encryption algorithm.

3. Create a custom encryption key using the Java key tool.

By accomplishing these tasks, students will grasp the nuances of stream cipher encryption using RC4, the workings of the Blowfish block cipher, and the process of generating secure encryption keys using Java's native tools.

**OUTCOME OF THE EXPERIMENT:**

Upon the successful execution of this Java program:

1. Participants will be adept at implementing the RC4 encryption technique using Java's cryptographic libraries.

2. Users will have encrypted the "Hello world" plaintext using the Blowfish algorithm and decrypted it back to retrieve the original message.

3. Students will have generated their custom encryption key using the Java key tool, bolstering their understanding of key management in cryptographic applications.

4. By juxtaposing RC4 and Blowfish, learners will appreciate the distinctions and applicability of stream ciphers versus block ciphers.

5. Participants will be better versed in Java's cryptographic capabilities, streamlining secure application development.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**RC4 Algorithm:**

1. **Key Setup:**

* Initialize two arrays, S and T, each containing values from 0 to 255.
* Permute the elements of array S based on the key bytes using the Key-Scheduling Algorithm (KSA).
* The key bytes are typically XORed with the array elements to shuffle them.

1. **Pseudorandom Generation Algorithm (PRGA):**

* Initialize two indices, i and j, both starting from 0.
* For each byte of the plaintext (or any data to be encrypted), generate a pseudorandom byte using the PRGA:
* Increment index i.
* Perform a swap between S[i] and S[j] where j = (j + S[i]) % 256.
* Calculate the sum k = S[i] + S[j] modulo 256.
* Generate a pseudorandom byte by indexing S with k and XORing it with the plaintext byte.

1. **Encryption and Decryption:**

* To encrypt plaintext, XOR the plaintext bytes with the pseudorandom bytes generated by the PRGA.
* Decryption is performed by XORing the ciphertext with the same pseudorandom bytes.

**PSEUDOCODE:**

import java.io.\*;

class RC4 {

public static void main(String args[]) throws IOException {

int temp = 0;

String ptext;

String key;

int s[] = new int[256];

int k[] = new int[256];

DataInputStream in = new DataInputStream(System.in);

System.out.print("ENTER PLAIN TEXT:\t");

ptext = in.readLine();

System.out.print("ENTER KEY TEXT:\t\t");

key = in.readLine();

char ptextc[] = ptext.toCharArray();

char keyc[] = key.toCharArray();

int cipher[] = new int[ptext.length()];

int decrypt[] = new int[ptext.length()];

int ptexti[] = new int[ptext.length()];

int keyi[] = new int[key.length()];

for (int i = 0; i < ptext.length(); i++) {

ptexti[i] = (int) ptextc[i];

}

for (int i = 0; i < key.length(); i++) {

keyi[i] = (int) keyc[i];

}

for (int i = 0; i < 255; i++) {

s[i] = i;

k[i] = keyi[i % key.length()];

}

int j = 0;

for (int i = 0; i < 255; i++) {

j = (j + s[i] + k[i]) % 256;

temp = s[i];

s[i] = s[j];

s[j] = temp;

}

int i = 0;

j = 0;

int z = 0;

for (int l = 0; l < ptext.length(); l++) {

i = (l + 1) % 256;

j = (j + s[i]) % 256;

temp = s[i];

s[i] = s[j];

s[j] = temp;

z = s[(s[i] + s[j]) % 256];

cipher[l] = z ^ ptexti[l];

decrypt[l] = z ^ cipher[l];

}

System.out.print("\nENCRYPTED:\t\t");

display(cipher);

System.out.print("\nDECRYPTED:\t\t");

display(decrypt);

}

static void display(int disp[]) {

char convert[] = new char[disp.length];

for (int l = 0; l < disp.length; l++) {

convert[l] = (char) disp[l];

System.out.print(convert[l]);

}

}

}

**OUTPUT:**

**ENTER PLAIN TEXT: This text is plain.**

**ENTER KEY TEXT: This text is a key.**

**ENCRYPTED: V¡u¬s8ÈSç°Û·P**

**DECRYPTED: This text is plain.**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Explain the primary differences between stream ciphers like RC4 and block ciphers like

Blowfish.

2. Why is key management crucial in cryptographic applications?

3. What vulnerabilities did the RC4 cipher historically face, leading to its decreased

popularity?

4. Describe the function of S-boxes in the Blowfish algorithm.

5. How does the Java key tool enhance the security of Java-based applications?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. What are the main components of the RC4 algorithm, and how do they contribute to the

encryption process?

2. How does the key size affect the strength of encryption in algorithms like Blowfish?

3. Why is it essential to use a secure method, like the Java key tool, to generate cryptographic keys?

4. Can you explain the "bit-flipping" vulnerability historically associated with RC4 in certain applications?

5. Describe the role of the permutation array (or state array) in the RC4 algorithm.

6. How does Blowfish ensure data security through its iterative rounds and dynamic S-boxes?

7. In what scenarios would you recommend using RC4, given its historical vulnerabilities?

8. Explain the importance of securely storing and managing keys generated by tools like the Java key tool.

9. How does the key expansion process in Blowfish contribute to its robustness against brute-force attacks?

10. Given the cryptographic capabilities of Java, how would you secure data transmission in a Java-based application?

**EXPERIMENT / PRACTICAL -8**

**Write a Java program to implement RSA algorithm.**

**OBJECTIVE OF THE EXPERIMENT:**

The primary objective of this Java program is to implement the RSA (Rivest-Shamir-Adleman) algorithm, one of the first practical public-key cryptosystems and widely used for secure data transmission. Through this program, students will learn the principles of asymmetric encryption, prime number generation, modular arithmetic, and key generation, enabling secure communications in a plethora of digital platforms

**OUTCOME OF THE EXPERIMENT:**

Upon the successful execution of this Java program:

1. Participants will have a clear understanding of the RSA algorithm, its public and private key generation mechanisms, and encryption and decryption processes.

2. Users will be able to encrypt plaintext messages using the RSA algorithm's public key and subsequently decrypt the ciphertext using the corresponding private key.

3. Students will appreciate the significance of prime numbers and their role in RSA's security.

4. Learners will understand the strengths and potential vulnerabilities of the RSA algorithm, fostering its effective utilization.

5. Through hands-on coding, students will enhance their programming skills in Java, particularly around big integer operations, modular arithmetic, and the intricacies of the RSA algorithm.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

1. **Key Generation:**

* Choose two distinct large prime numbers, p and q.
* Compute the modulus n = p \* q.
* Compute the totient function value φ(n) = (p - 1) \* (q - 1).
* Choose an integer e (the public exponent) such that 1 < e < φ(n) and gcd(e, φ(n)) = 1. Common values for e include 3 and 65537.
* Compute the private exponent d (the modular multiplicative inverse of e modulo φ(n)), i.e., d \* e ≡ 1 (mod φ(n)).

1. **Key Distribution:**

* The public key consists of n and e and is made available to everyone. It can be used for encryption.
* The private key consists of n and d and must be kept secret. It is used for decryption and digital signing.

1. **Encryption:**

* The sender converts the plaintext message into a numerical value M using a reversible encoding (e.g., converting characters to their ASCII/Unicode values).
* The sender computes the ciphertext C as C ≡ M^e (mod n) using the recipient's public key.

1. **Decryption:**

* The recipient uses their private key to compute the original message value M as M ≡ C^d (mod n)

**PSEUDOCODE:**

import java.util.\*;

import java.math.\*;

class RSA

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int p,q,n,z,d=0,e,i;

System.out.println("Enter the number to be encrypted and decrypted");

int msg=sc.nextInt();

double c;

BigInteger msgback;

System.out.println("Enter 1st prime number p");

p=sc.nextInt();

System.out.println("Enter 2nd prime number q");

q=sc.nextInt();

n=p\*q;

z=(p-1)\*(q-1);

System.out.println("the value of z = "+z);

for(e=2;e<z;e++)

{

if(gcd(e,z)==1) // e is for public key exponent

{

break;

}

}

System.out.println("the value of e = "+e);

for(i=0;i<=9;i++)

{

int x=1+(i\*z);

if(x%e==0) //d is for private key exponent

{

d=x/e;

break;

}

}

System.out.println("the value of d = "+d);

c=(Math.pow(msg,e))%n;

System.out.println("Encrypted message is : -");

System.out.println(c);

//converting int value of n to BigInteger

BigInteger N = BigInteger.valueOf(n);

//converting float value of c to BigInteger

BigInteger C = BigDecimal.valueOf(c).toBigInteger();

msgback = (C.pow(d)).mod(N);

System.out.println("Derypted message is : -");

System.out.println(msgback);

}

static int gcd(int e, int z)

{

if(e==0)

return z;

else

return gcd(z%e,e);

}

}

**OUTPUT:**

**Enter the number to be encrypted and decrypted**

**55**

**Enter 1st prime number p**

**13**

**Enter 2nd prime number q**

**23**

**the value of z = 264**

**the value of e = 5**

**the value of d = 53**

**Encrypted message is : -**

**100.0**

**Derypted message is : -**

**55**

**PRE LAB QUESTIONS**

Here are pre-lab questions for the given program that computes the objective of the experiment:

1. How does the RSA algorithm ensure both confidentiality and authenticity in data

transmission?

2. Explain the roles of the public key and private key in the RSA algorithm.

3. What is the significance of Euler's totient function in the RSA algorithm?

4. Why is the selection of large prime numbers critical for the security of the RSA algorithm?

5. Describe the basic steps involved in RSA's encryption and decryption processes.

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. Why is RSA considered an asymmetric cryptographic algorithm, and how does it differ

from symmetric algorithms?

2. How do prime factorization problems underpin the security of RSA?

3. Describe the potential impact of quantum computing on the RSA algorithm's security.

4. Why is padding essential when encrypting messages using RSA?

5. What vulnerabilities arise if small prime numbers are used in the RSA algorithm?

6. How do public key infrastructures (PKIs) enhance the practical usability of RSA in real-world applications?

7. Describe the role of the Chinese Remainder Theorem in optimizing RSA decryption.

8. In RSA, how do you determine a suitable public exponent, and why is the number 65537 commonly chosen?

9. Explain the concept of "digital signatures" in RSA and their significance.

10. Considering the computational intensity of RSA, in what scenarios would you recommend its use over other encryption algorithms?

**EXPERIMENT / PRACTICAL -9**

**Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this experiment is to understand the Diffie-Hellman Key Exchange mechanism, which allows two parties to establish a shared secret over an insecure channel. By using HTML and JavaScript, students will get hands-on experience in web-based cryptography, ensuring secure communications in a web environment.

**OUTCOME OF THE EXPERIMENT:**

After completing this experiment:

1. Participants will have a clear understanding of the Diffie-Hellman Key Exchange mechanism and its significance in secure communications.

2. Students will be able to implement the Diffie-Hellman process using JavaScript, facilitating the generation of a shared secret between two parties.

3. Learners will comprehend the importance of prime numbers and primitive roots in the key exchange process.

4. Through a web interface in HTML, participants will visually experience the steps of the Diffie-Hellman process.

5. The students will have enhanced their skills in web-based cryptographic implementations, bridging the gap between cryptographic theory and web application security.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

1. **Initialization:**

* Select a prime number p and a primitive root modulo p, denoted as g.
* Both parties agree on the values of p and g.

1. **Key Generation:**

* Each party (Alice and Bob) independently selects a private key:
* Alice chooses a private key a (randomly selected from a certain range).
* Bob chooses a private key b (also randomly selected).

1. **Public Key Calculation:**

* Each party calculates their public key:
* Alice computes A = g^a mod p and sends A to Bob.
* Bob computes B = g^b mod p and sends B to Alice.

1. **Shared Secret Calculation:**

* Both parties independently calculate the shared secret key using the received public keys:
* Alice calculates s = B^a mod p.
* Bob calculates s = A^b mod p.

1. **Result:**

* Both Alice and Bob have now arrived at the same shared secret key s, which can be used for further symmetric encryption or other cryptographic purposes.

**PSEUDOCODE:**

//HTML CODE

<!DOCTYPE html>

<html>

<head>

<title>Input Form Example</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #f0f0f0;

}

.container {

background-color: #fff;

border-radius: 8px;

padding: 20px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

width: 360px;

}

label {

display: block;

margin-bottom: 5px;

color: #555;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 10px;

margin-right: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

padding: 10px 15px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

width: 100%;

}

button:hover {

background-color: #0056b3;

}

p {

margin-top: 20px;

color: #333;

}

</style>

<script>

function power(a, b, p) {

if (b == 1)

return a;

else

return((Math.pow(a, b)) % p);

}

function diffie\_hellman() {

// This program calculates the Key for two persons using the Diffie-Hellman Key exchange algorithm

var P, G, x, a, y, b, ka, kb;

// Both the persons will be agreed upon the

// public keys G and P

// A prime number P is taken

P = document.getElementById("field1").value;

// A primitive root for P, G is taken

G = document.getElementById("field2").value;

// User1 will choose the private key a

// a is the chosen private key

a = document.getElementById("field3").value;

// User2 will choose the private key b

// b is the chosen private key

b = document.getElementById("field4").value;

// Gets the generated key

x = power(G, a, P);

y = power(G, b, P);

// Generating the secret key after the exchange of keys

ka = power(y, a, P); // Secret key for User1

kb = power(x, b, P); // Secret key for User2

document.getElementById("output1").innerHTML = ka

document.getElementById("output2").innerHTML = kb

}

</script>

</head>

<body>

<div class="container">

<h2 style="text-align: center;">Diffie Hellman</h2>

<label for="field1">Value of P:</label>

<input type="text" id="field1" required>

<label for="field2">Value of G(Primitive root of P):</label>

<input type="text" id="field2" required>

<label for="field3">Private key a for user1:</label>

<input type="text" id="field3" required>

<label for="field4">Private key b for user2:</label>

<input type="text" id="field4" required>

<button id="submitButton" onclick="diffie\_hellman()">Submit</button>

<p>Secret key for the User1: <span id="output1"></p></p>

<p>Secret key for the User2: <span id="output2"></p></p>

</div>

</body>

</html>

**//Java Code**import java.util.\*;

public class DiffieHellman {

// main() method start

public static void main(String[] args)

{

long P, G, x, a, y, b, ka, kb;

// create Scanner class object to take input from user

Scanner sc = new Scanner(System.in);

System.out.println("Both the users should be agreed upon the public keys G and P");

// take inputs for public keys from the user

System.out.println("Enter value for public key G:");

G = sc.nextLong();

System.out.println("Enter value for public key P:");

P = sc.nextLong();

// get input from user for private keys a and b selected by User1 and User2

System.out.println("Enter value for private key a selected by user1:");

a = sc.nextLong();

System.out.println("Enter value for private key b selected by user2:");

b = sc.nextLong();

// call calculatePower() method to generate x and y keys

x = calculatePower(G, a, P);

y = calculatePower(G, b, P);

// call calculatePower() method to generate ka and kb secret keys after the exchange of x and y keys

// calculate secret key for User1

ka = calculatePower(y, a, P);

// calculate secret key for User2

kb = calculatePower(x, b, P);

// print secret keys of user1 and user2

System.out.println("Secret key for User1 is:" + ka);

System.out.println("Secret key for User2 is:" + kb);

}

// create calculatePower() method to find the value of x ^ y mod P

private static long calculatePower(long x, long y, long P)

{

long result = 0;

if (y == 1){

return x;

}

else{

result = ((long)Math.pow(x, y)) % P;

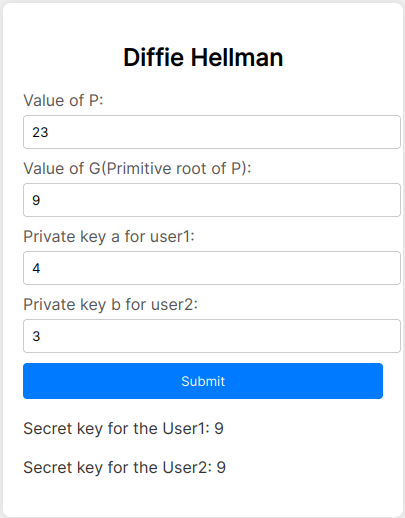
return result;

}

}

}

**OUTPUT:**

****

**JAVA OUTPUT:**

**Both the users should be agreed upon the public keys G and P**

**Enter value for public key G:**

**9**

**Enter value for public key P:**

**23**

**Enter value for private key a selected by user1:**

**4**

**Enter value for private key b selected by user2:**

**3**

**Secret key for User1 is:9**

**Secret key for User2 is:9**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Describe the significance of the Diffie-Hellman Key Exchange mechanism in cryptography.

2. How does Diffie-Hellman ensure that an eavesdropper cannot easily determine the shared secret, even if they observe the public key exchange process?

3. What is the role of prime numbers in the Diffie-Hellman process?

4. Why is the Diffie-Hellman method considered a "key exchange" method and not an encryption method?

5. Explain the potential vulnerabilities of the Diffie-Hellman Key Exchange.

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. How does the choice of the primitive root affect the security of the Diffie-Hellman process?

2. Can Diffie-Hellman be used for message authentication? Why or why not?

3. What happens if two parties using Diffie-Hellman choose poor or small private keys?

4. How can the Diffie-Hellman Key Exchange mechanism be combined with other encryption mechanisms for secure communication?

5. Explain the concept of "perfect forward secrecy" in the context of Diffie-Hellman.

**EXPERIMENT / PRACTICAL -10**

**Calculate the message digest of a text using the SHA-1 algorithm in JAVA.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this Java program is to compute the message digest of a given text using the SHA-1 (Secure Hash Algorithm 1) algorithm. By doing so, students will understand how cryptographic hash functions operate, the significance of data integrity, and the importance of message digests in validating data authenticity.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion of this Java program:

1. Users will comprehend the workings of the SHA-1 algorithm and how it produces a fixed-size hash value from input data of any size.

2. Participants will be able to generate the SHA-1 hash of any text, gaining a practical insight into data integrity checks.

3. Learners will recognize the uniqueness and determinism of hash values produced by cryptographic hash functions.

4. Students will appreciate the applications of SHA-1, including password storage, data integrity checks, and digital signatures, despite its modern vulnerabilities.

5. Through coding in Java, students will deepen their understanding of Java's cryptographic libraries and how they can be leveraged for various security operations.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

1. **Padding:**

* Append a '1' bit to the message.
* Append '0' bits to the message until the length is congruent to 448 modulo 512.
* Append the original message length (in bits) as a 64-bit big-endian integer.

1. **Message Breakdown:**

* Divide the padded message into 512-bit blocks.

1. **Initial Hash Values:**

* Initialize five 32-bit variables (H0, H1, H2, H3, H4) with the initial hash values.

1. **Process Blocks:** For each block:

* Prepare sixteen 32-bit words (W0, W1, ..., W15) from the 512-bit block.
* Extend the 16 words into 80 words.
* Initialize temporary variables (A, B, C, D, E).
* Perform 80 rounds of processing:
* Determine the function for the round based on the round number (Ch, Parity, Maj, etc.).
* Calculate new values for A, B, C, D, E using bitwise operations and constants.
* Update the hash values H0, H1, H2, H3, H4 with the new A, B, C, D, E values.

1. **Final Hash:**

* Concatenate the hash values H0, H1, H2, H3, H4 to obtain the final SHA-1 hash value.

**PSEUDOCODE:**

public class SHA1 {

public static void main(String[] args) {

String input = "Hello, world!";

String sha1Hash = sha1(input);

System.out.println("Input: " + input);

System.out.println("SHA-1 hash: " + sha1Hash);

}

public static String sha1(String input) {

int[] h0 = {0x67452301};

int[] h1 = {0xEFCDAB89};

int[] h2 = {0x98BADCFE};

int[] h3 = {0x10325476};

int[] h4 = {0xC3D2E1F0};

byte[] data = input.getBytes();

int[] paddedData = padData(data);

for (int i = 0; i < paddedData.length; i += 64) {

int[] words = new int[80];

for (int j = 0; j < 16; j++) {

words[j] = (paddedData[i + j \* 4] << 24) |

(paddedData[i + j \* 4 + 1] << 16) |

(paddedData[i + j \* 4 + 2] << 8) |

(paddedData[i + j \* 4 + 3]);

}

for (int j = 16; j < 80; j++) {

words[j] = leftRotate(words[j - 3] ^ words[j - 8] ^ words[j - 14] ^ words[j - 16], 1);

}

int a = h0[0];

int b = h1[0];

int c = h2[0];

int d = h3[0];

int e = h4[0];

for (int j = 0; j < 80; j++) {

int f, k;

if (j < 20) {

f = (b & c) | (~b & d);

k = 0x5A827999;

} else if (j < 40) {

f = b ^ c ^ d;

k = 0x6ED9EBA1;

} else if (j < 60) {

f = (b & c) | (b & d) | (c & d);

k = 0x8F1BBCDC;

} else {

f = b ^ c ^ d;

k = 0xCA62C1D6;

}

int temp = leftRotate(a, 5) + f + e + k + words[j];

e = d;

d = c;

c = leftRotate(b, 30);

b = a;

a = temp;

}

h0[0] += a;

h1[0] += b;

h2[0] += c;

h3[0] += d;

h4[0] += e;

}

String sha1Hash = toHexString(h0) + toHexString(h1) + toHexString(h2) + toHexString(h3) + toHexString(h4);

return sha1Hash;

}

private static int leftRotate(int value, int shift) {

return (value << shift) | (value >>> (32 - shift));

}

private static int[] padData(byte[] data) {

int dataLength = data.length;

int paddedLength = ((dataLength + 8) / 64 + 1) \* 64;

int[] paddedData = new int[paddedLength];

for (int i = 0; i < dataLength; i++) {

paddedData[i] = data[i] & 0xFF;

}

paddedData[dataLength] = 0x80;

long bitLength = dataLength \* 8L;

paddedData[paddedLength - 8] = (int) (bitLength >>> 56);

paddedData[paddedLength - 7] = (int) (bitLength >>> 48);

paddedData[paddedLength - 6] = (int) (bitLength >>> 40);

paddedData[paddedLength - 5] = (int) (bitLength >>> 32);

paddedData[paddedLength - 4] = (int) (bitLength >>> 24);

paddedData[paddedLength - 3] = (int) (bitLength >>> 16);

paddedData[paddedLength - 2] = (int) (bitLength >>> 8);

paddedData[paddedLength - 1] = (int) bitLength;

return paddedData;

}

private static String toHexString(int[] value) {

StringBuilder sb = new StringBuilder();

for (int v : value) {

sb.append(String.format("%08x", v));

}

return sb.toString();

}

}

**OUTPUT:**

**Input: Hello, world!**

**SHA-1 hash: 943a702d06f34599aee1f8da8ef9f7296031d699**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. What is a cryptographic hash function, and how is it different from other functions?

2. Describe the characteristics of a secure hash function.

3. How does the SHA-1 algorithm generate a fixed-size hash (160 bits) irrespective of the

input size?

4. What are the potential vulnerabilities of SHA-1, and how have they impacted its use in

modern systems?

5. Why is it nearly impossible to produce the original data from its SHA-1 hash?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. How does the concept of "collision resistance" apply to SHA-1?

2. Describe a real-world application where SHA-1 might still be in use, and discuss its

implications.

3. Given the vulnerabilities of SHA-1, which other hashing algorithms would you

recommend for more secure applications?

4. How does the speed of generating a hash influence the choice of a hashing algorithm in

various applications?

5. Why is the SHA-1 hash popularly used in Git for version control, and what are the

potential risks?

**EXPERIMENT / PRACTICAL -11**

**Calculate the message digest of a text using the MD5 algorithm in JAVA.  
  
OBJECTIVE OF THE EXPERIMENT:**

The objective of this Java program is to compute the message digest of a given text using the MD5 (Message Digest Algorithm 5) algorithm. By performing this exercise, students will understand the principles behind cryptographic hash functions, learn about ensuring data integrity, and grasp the concepts of creating fixed-length hash values from variable-length input data.

**OUTCOME OF THE EXPERIMENT:**

Upon successfully executing this Java program:

1. Users will have gained insight into the MD5 algorithm and its mechanism of generating a fixed-size 128-bit hash value.

2. Students will be capable of producing the MD5 hash of any given text, thereby understanding practical aspects of data verification and integrity assurance.

3. Participants will understand the determinism of cryptographic hash functions, where a specific input always produces the same hash.

4. Learners will be introduced to the vulnerabilities and criticisms of the MD5 algorithm and its decreasing security in modern cryptographic applications.

5. Through Java programming, students will experience firsthand the cryptographic libraries of Java and how they can be utilized to implement various security functionalities.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**MD5 Hash Function Algorithm:**

1. **Padding:**

* Append a '1' bit to the message.
* Append '0' bits to the message until the length is congruent to 448 modulo 512.
* Append the original message length (in bits) as a 64-bit little-endian integer.

1. **Message Breakdown:**

* Divide the padded message into 512-bit blocks.

1. **Initial Hash Values:**

* Initialize four 32-bit variables (A, B, C, D) with the initial hash values.

1. **Process Blocks:** For each block:

* Prepare sixteen 32-bit words (W0, W1, ..., W15) from the 512-bit block.
* Extend the 16 words into 64 words using a mathematical function.
* Initialize temporary variables (A', B', C', D').
* Perform 64 rounds of processing:
* Determine the function for the round based on the round number (F, G, H, I).
* Calculate new values for A', B', C', D' using bitwise operations and constants.
* Update A, B, C, D with the new A', B', C', D' values.

1. **Final Hash:**

* Concatenate the four hash values A, B, C, D to obtain the final MD5 hash value.

**PSEUDOCODE:**

public class MD5 {

private static final int INIT\_A = 0x67452301;

private static final int INIT\_B = (int) 0xEFCDAB89L;

private static final int INIT\_C = (int) 0x98BADCFEL;

private static final int INIT\_D = 0x10325476;

private static final int[] SHIFT\_AMTS = { 7, 12, 17, 22, 5, 9, 14, 20, 4, 11, 16, 23, 6, 10, 15, 21 };

private static final int[] TABLE\_T = new int[64];

static {

for (int i = 0; i < 64; i++)

TABLE\_T[i] = (int) (long) ((1L << 32) \* Math.abs(Math.sin(i + 1)));

}

public static byte[] computeMD5(byte[] message) {

int messageLenBytes = message.length;

int numBlocks = ((messageLenBytes + 8) >>> 6) + 1;

int totalLen = numBlocks << 6;

byte[] paddingBytes = new byte[totalLen - messageLenBytes];

paddingBytes[0] = (byte) 0x80;

long messageLenBits = (long) messageLenBytes << 3;

for (int i = 0; i < 8; i++) {

paddingBytes[paddingBytes.length - 8 + i] = (byte) messageLenBits;

messageLenBits >>>= 8;

}

int a = INIT\_A;

int b = INIT\_B;

int c = INIT\_C;

int d = INIT\_D;

int[] buffer = new int[16];

for (int i = 0; i < numBlocks; i++) {

int index = i << 6;

for (int j = 0; j < 64; j++, index++)

buffer[j >>> 2] = ((int) ((index < messageLenBytes) ? message[index] : paddingBytes[index - messageLenBytes]) << 24) | (buffer[j >>> 2] >>> 8);

int originalA = a;

int originalB = b;

int originalC = c;

int originalD = d;

for (int j = 0; j < 64; j++) {

int div16 = j >>> 4;

int f = 0;

int bufferIndex = j;

switch (div16) {

case 0:

f = (b & c) | (~b & d);

break;

case 1:

f = (b & d) | (c & ~d);

bufferIndex = (bufferIndex \* 5 + 1) & 0x0F;

break;

case 2:

f = b ^ c ^ d;

bufferIndex = (bufferIndex \* 3 + 5) & 0x0F;

break;

case 3:

f = c ^ (b | ~d);

bufferIndex = (bufferIndex \* 7) & 0x0F;

break;

}

int temp = b + Integer.rotateLeft(a + f + buffer[bufferIndex] + TABLE\_T[j], SHIFT\_AMTS[(div16 << 2) | (j & 3)]);

a = d;

d = c;

c = b;

b = temp;

}

a += originalA;

b += originalB;

c += originalC;

d += originalD;

}

byte[] md5 = new byte[16];

int count = 0;

for (int i = 0; i < 4; i++) {

int n = (i == 0) ? a : ((i == 1) ? b : ((i == 2) ? c : d));

for (int j = 0; j < 4; j++) {

md5[count++] = (byte) n;

n >>>= 8;

}

}

return md5;

}

public static String toHexString(byte[] b) {

StringBuilder sb = new StringBuilder();

for (int i = 0; i < b.length; i++) {

sb.append(String.format("%02X", b[i] & 0xFF));

}

return sb.toString();

}

public static void main(String[] args) {

String[] testStrings = { "Hello, world!", "Sanfoundry", "Message Digest", "abcdefghijklmnopqrstuvwxyz" };

for (String s : testStrings)

System.out.println("0x" + toHexString(computeMD5(s.getBytes())) + " <== \"" + s + "\"");

}

}

**OUTPUT:**

**0x6CD3556DEB0DA54BCA060B4C39479839 <== "Hello, world!"**

**0x123EC1617559F98A4C86AF629FEF21E6 <== "Sanfoundry"**

**0xBBD9D8CC4AD8AD2599DBF623E7E5282E <== "Message Digest"**

**0xC3FCD3D76192E4007DFB496CCA67E13B <== "abcdefghijklmnopqrstuvwxyz"**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. What are the primary features and characteristics of a cryptographic hash function?

2. Describe the process by which the MD5 algorithm produces a 128-bit hash value.

3. Discuss the known vulnerabilities associated with the MD5 hashing algorithm.

4. How does the MD5 hash ensure data integrity?

5. Why is it theoretically challenging to derive the original data from its MD5 hash?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. What is a "collision" in the context of cryptographic hashing, and why is it a concern for

MD5?

1. Explain why MD5 is considered a broken and vulnerable hashing algorithm in modern

cryptography.

3. How can "salting" be used in conjunction with MD5, and what benefits does it provide?

4. If MD5 is considered vulnerable, why is it still in use in certain systems and applications?

5. What are some modern alternatives to MD5 that offer stronger cryptographic security?

**EXPERIMENT / PRACTICAL -12**

**Implement Simple Columner Transposition technique and Advanced Columner Transposition technique.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this exercise is to implement both the Simple Columnar Transposition technique and the Advanced Columnar Transposition technique. These are methods of symmetric encryption that involve rearranging plaintext characters based on a key. By studying these techniques, students will understand the foundations of transposition ciphers and the importance of key-based encryption and decryption.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion:

1. Participants will understand the principles behind transposition ciphers.

2. They will be proficient in performing encryption and decryption using both Simple and Advanced Columnar Transposition techniques.

3. Students will understand the significance of keys in determining the encryption pattern.

4. They will recognize the differences between substitution and transposition ciphers.

5. Students will gain hands-on experience in symmetric encryption techniques.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:  
 javac my\_program.java  
 java my\_program  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**Columnar:**

**Encryption**

In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.

1. The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
2. Width of the rows and the permutation of the columns are usually defined by a keyword.
3. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be “3 1 2 4”.
4. Any spare spaces are filled with nulls or left blank or placed by a character (Example: \_).
5. Finally, the message is read off in columns, in the order specified by the keyword.

**Decryption**

1. To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length.
2. Then, write the message out in columns again, then re-order the columns by reforming the key word.

## Advanced Columnar:

For encryption, perform Columnar encryption 'n' times, where n is the number of iterations provided by the user.

Similarly, decryption has to be performed 'n' times using Columnar decryption function.

**PSEUDOCODE:**

//Columnar Program

import java.util.Arrays;

public class Columnar {

public static void main(String[] args) {

String msg = "Geeks for Geeks";

String key = "HACK";

String cipher = encrypt(msg, key);

System.out.println("Encrypted Message: " + cipher);

String decryptedMessage = decrypt(cipher, key);

System.out.println("Decrypted Message: " + decryptedMessage);

}

public static String encrypt(String msg, String key) {

StringBuilder cipher = new StringBuilder();

int kIndex = 0;

double msgLen = msg.length();

char[] msgArr = msg.toCharArray();

char[] keyArr = key.toCharArray();

char[] sortedKey = key.toCharArray();

Arrays.sort(sortedKey);

int col = key.length();

int row = (int) Math.ceil(msgLen / col);

int fillNull = (row \* col) - msg.length();

char[] paddedMsg = new char[(int) (msgLen + fillNull)];

System.arraycopy(msgArr, 0, paddedMsg, 0, msgArr.length);

Arrays.fill(paddedMsg, msgArr.length, paddedMsg.length, '-');

char[][] matrix = new char[row][col];

for (int i = 0; i < msgLen + fillNull; i += col) {

matrix[i / col] = Arrays.copyOfRange(paddedMsg, i, i + col);

}

for (int i = 0; i < col; i++) {

int currIdx = key.indexOf(sortedKey[kIndex]);

for (char[] rowArray : matrix) {

cipher.append(rowArray[currIdx]);

}

kIndex++;

}

return cipher.toString();

}

public static String decrypt(String cipher, String key) {

int kIndex = 0;

int msgIndex = 0;

double msgLen = cipher.length();

char[] cipherArr = cipher.toCharArray();

char[] keyArr = key.toCharArray();

char[] sortedKey = key.toCharArray();

Arrays.sort(sortedKey);

int col = key.length();

int row = (int) Math.ceil(msgLen / col);

char[][] decCipher = new char[row][col];

for (int i = 0; i < col; i++) {

int currIdx = key.indexOf(sortedKey[kIndex]);

for (int j = 0; j < row; j++) {

decCipher[j][currIdx] = cipherArr[msgIndex];

msgIndex++;

}

kIndex++;

}

StringBuilder msg = new StringBuilder();

for (char[] rowArray : decCipher)

for (char c : rowArray)

msg.append(c);

return msg.toString();

}

}

**OUTPUT:**

**PlainText: Hello, CSE5!**

**Key: HACK**

**Encrypted Message: e,El 5HoSlC!**

**Decrypted Message: Hello, CSE5!**

**//Advanced Columnar Program**

class Adv\_Columnar {

public static void main(String[] args) {

Columnar c = new Columnar();

String message = "Hello Geek";

String key = "megabuck";

int iterations = 4;

Adv\_Columnar advC = new Adv\_Columnar();

String cipher = advC.advColumnarEncrypt(message, key, iterations);

String decrypted = advC.advColumnarDecrypt(cipher, key, iterations);

System.out.println("Message: " + message);

System.out.println("Key: " + key);

System.out.println("Iterations: " + iterations);

System.out.println("CipherText: " + cipher);

System.out.println("Decrypted: " + decrypted);

}

String advColumnarEncrypt(String message, String key, int iterations) {

String cipher = message;

for (int i=0; i<iterations; i++)

cipher = Columnar.encrypt(cipher, key);

return cipher;

}

String advColumnarDecrypt(String cipher, String key, int iterations) {

String message = cipher;

for (int i=0; i<iterations; i++)

message = Columnar.decrypt(message, key);

return message;

}

}

**OUTPUT:**

**Message: Hello CSE5**

**Key: megabuck**

**Iterations: 9**

**CipherText: H-olC-eSl5---E -**

**Decrypted: Hello CSE5------**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Describe the Simple Columnar Transposition technique. How does it differ from Advanced Columnar Transposition?

2. Why is the key essential in Columnar Transposition techniques?

3. How does a Columnar Transposition cipher ensure the confidentiality of information?

4. In what scenarios is the Columnar Transposition technique most effective?

5. What are the potential vulnerabilities of the Columnar Transposition techniques?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. How would you decrypt a message encrypted using the Simple Columnar Transposition

technique?

2. Explain the significance of the key's length in the Columnar Transposition cipher.

3. How does the Advanced Columnar Transposition technique handle keys that have

repeating characters?

4. In the context of transposition ciphers, what is the difference between a key and a

passphrase?

5. How can the Columnar Transposition techniques be combined with other cryptographic

methods to enhance security?

**EXPERIMENT / PRACTICAL -13**

**Implement** **Euclidean Algorithm and Advanced Euclidean Algorithm.**

**OBJECTIVE OF THE EXPERIMENT:**

The objective of this exercise is to understand and implement the Euclidean Algorithm, which is used to find the Greatest Common Divisor (GCD) of two numbers, and the Extended Euclidean Algorithm, which finds the multiplicative inverse of a number in modular arithmetic. These algorithms lay the foundation for numerous cryptographic techniques, especially in public key cryptography.

**OUTCOME OF THE EXPERIMENT:**

Upon successful completion:

1. Participants will have a clear understanding of the Euclidean and Extended Euclidean Algorithms and their significance.

2. They will be proficient in determining the GCD of two integers using the Euclidean Algorithm.

3. Students will be capable of computing the multiplicative inverse of an integer using the Extended Euclidean Algorithm.

4. Learners will appreciate the role of these algorithms in cryptographic techniques, especially RSA encryption.

5. Students will have enhanced their problem-solving abilities with algorithmic mathematics through practical implementation.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**Euclidean Algorithm :**

**Input:**

* Given a, b (positive integers, where a >= b).

**Step 1: Divide and Find Remainder:**

* Calculate quotient q and remainder r:
* q = floor(a / b)
* r = a % b

**Step 2: Check Remainder:**

* If r is zero, go to Step 5.
* If r is not zero, continue to Step 4.

**Step 3: Update and Repeat:**

* Update a to be b, and b to be r.
* Repeat from Step 1.

**Output:**

* The GCD of a and b is b.

Advanced **Euclidean Algorithm :**

**Input:**

* Given two positive integers, a and b, where a >= b.

**Step 1: Initialize Coefficients:**

* Initialize four variables: x1, y1, x2, y2.
* Set x1 = 1, y1 = 0, x2 = 0, and y2 = 1.

**Step 2: Loop until Remainder is Zero:**

* While b is not zero:
* Calculate quotient q and remainder r:
* q = floor(a / b)
* r = a % b
* Update a to be b, and b to be r.
* Update x1, y1, x2, and y2 using a temporary variable tmp:
* tmp = x2, x2 = x1 - q \* x2, x1 = tmp
* tmp = y2, y2 = y1 - q \* y2, y1 = tmp

**Step 3: Output Coefficients:**

* The coefficients x1 and y1 of Bézout's identity represent the GCD of a and b as a linear combination of a and b, i.e., GCD(a, b) = a \* x1 + b \* y1.

**Output:**

* The GCD of the original values a and b is the value of a when the algorithm terminates.
* The coefficients x1 and y1 are also obtained, providing Bézout's identity.

**PSEUDOCODE:**

// Java program to demonstrate working of extended

// Euclidean Algorithm

class Euclid {

// Euclidean Algorithm for GCD

public static int gcdExtended(int a, int b, int x,

int y)

{

// Base Case

if (a == 0) {

x = 0;

y = 1;

return b;

}

int x1 = 1,

y1 = 1; // To store results of recursive call

int gcd = gcdExtended(b % a, a, x1, y1);

// Update x and y using results of recursive

// call

x = y1 - (b / a) \* x1;

y = x1;

return gcd;

}

// Driver Program

public static void main(String[] args)

{

int x = 1, y = 1;

int a = 35, b = 15;

int g = gcdExtended(a, b, x, y);

System.out.println("gcd(" + a + " , " + b

+ ") = " + g);

}

}

**OUTPUT:**

**Given inputs: 35, 15**

**gcd(35 , 15) = 5**

// Java program to demonstrate working of extended

// Euclidean Algorithm

class Euclid\_Extended {

static public void gcdExtended(long a, long b)

{

long x = 0, y = 1, lastx = 1, lasty = 0, temp;

while (b != 0)

{

long q = a / b;

long r = a % b;

a = b;

b = r;

temp = x;

x = lastx - q \* x;

lastx = temp;

temp = y;

y = lasty - q \* y;

lasty = temp;

}

System.out.println("GCD "+ a +" and its Roots x : "+ lastx +" y :"+ lasty);

}

// Driver Program

public static void main(String[] args)

{

long a = 35, b = 15;

//this will print result like

//Roots x : 1 y :-2

gcdExtended(a, b);

}

}

**OUTPUT:**

**Given a and b for ax+by = GCD(a,b) are:**

**a = 35 b = 15**

**GCD 5 and its Roots x : 1 y :-2**

**Final Equation:**

**1x + -2y = 5**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. What is the primary purpose of the Euclidean Algorithm? How does it achieve this

purpose?

2. How does the Extended Euclidean Algorithm extend the basic Euclidean Algorithm?

3. What role does the Greatest Common Divisor (GCD) play in number theory and

cryptography?

4. How is the multiplicative inverse different from the regular inverse?

5. Why is the multiplicative inverse essential in modular arithmetic and cryptographic

algorithms?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. Describe the primary steps of the Euclidean Algorithm.

2. How can the Extended Euclidean Algorithm be employed to solve Diophantine equations?

3. If two numbers have a GCD of 1, what special name is given to their relationship?

4. In the context of the RSA algorithm, why is the Extended Euclidean Algorithm crucial?

5. Can the Extended Euclidean Algorithm find multiplicative inverses in non-modular

arithmetic? Why or why not?

**EXPERIMENT / PRACTICAL -14**

**Familiarize the cryptographic tools (opencv).**

**OBJECTIVE OF THE EXPERIMENT:**

The main objective of this task is to acquaint learners with the integration of cryptographic tools using OpenCV (Open Source Computer Vision Library). While OpenCV is primarily known for its functionalities related to computer vision, it can be combined with cryptographic libraries to ensure the secure processing, storage, and transmission of images and video content.

**OUTCOME OF THE EXPERIMENT:**

By the end of this experiment:

1. Participants will be familiar with the basic functionalities of OpenCV and its applicability in cryptographic contexts.

2. Learners will understand how to handle image and video data securely using OpenCV in combination with cryptographic tools.

3. Students will recognize the importance of encrypting visual data, especially in the age of digital surveillance and data breaches.

4. Participants will gain hands-on experience with the encryption and decryption of images, understanding practical concerns like maintaining image quality post-decryption.

5. Through the merging of computer vision and cryptography, students will appreciate the interdisciplinary nature of modern computational problems.

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Why is it essential to secure visual data (like images and videos) in today's digital era?

2. How does encrypting an image differ from encrypting textual data?

3. What potential vulnerabilities might arise if visual data is not encrypted, especially in

applications like surveillance or personal photography?

4. How does the `cryptography` library in Python ensure the secure encryption and

decryption of data?

5. What are the advantages of combining a library like OpenCV with cryptographic tools?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:

1. Describe the basic principles of symmetric encryption.

2. How would you handle the encryption of real-time video streams using OpenCV and

cryptographic libraries?

3. What considerations should be kept in mind when encrypting visual data to ensure quality

is maintained post-decryption?

4. Are there any specific cryptographic algorithms you would recommend for image or video

encryption? Why?

5. How would you handle key management and distribution in a scenario where multiple

users need access to encrypted visual data?

**EXPERIMENT / PRACTICAL -15**

**Generate Key using One-Time pad to perform encryption and decryption using vernam cipher.**

**OBJECTIVE OF THE EXPERIMENT:**

The primary objective of this task is to implement the Vernam Cipher, commonly known as the One-Time Pad (OTP). The One-Time Pad is a type of symmetric encryption that, when used correctly, is considered unbreakable. It employs a truly random key that is as long as the message itself.

**OUTCOME OF THE EXPERIMENT:**

By the end of this coding exercise:

1. Participants will understand the principles and security of the Vernam Cipher or One-Time Pad.

2. Users will be able to encrypt and decrypt messages using a truly random key of equivalent length.

3. Students will appreciate the requirement for true randomness in the OTP and understand the vulnerabilities if this requirement is not met.

4. Participants will realize the challenges in key generation, distribution, and management for the OTP.

**SYSTEM REQUIREMENTS:**

To run a Java program that computes the objective of the experiment, you need to ensure that your system meets the following requirements:

1. **Java :**  Make sure you have Java installed on your system.
2. **Text Editor or Integrated Development Environment (IDE):** You need a text editor or IDE to write and edit your Java code. Popular choices include Visual Studio Code, Sublime Text, Atom.
3. **Operating System:** Java is compatible with various operating systems like Windows, macOS, and GNU/Linux. So, ensure that your system is running one of these supported operating systems.
4. **Hardware Requirements:** Java is lightweight and don't have significant hardware requirements. Any modern computer with sufficient memory and processing power should be capable of running Java programs.

Once your system meets these requirements, you can create a Java file (e.g., **my\_program.java**) and write the Java program to compute the objective of experiment, as shown in the procedure. Save the file and run it in the command prompt or IDE.

For example, if you are using the command prompt or terminal, navigate to the directory where your my\_program.java file is located and run the following command:

javac my\_program.java  
  
java my\_program  
  
If everything is set up correctly, the program will execute, and you should see the output.

**ALGORITHM / PROCEDURE:**

**One-Time Pad Algorithm:**

1. **Key Generation:**

* Generate a secret random key, which should be at least as long as the plaintext message.
* The key should consist of truly random characters (usually letters, digits, or other symbols).

1. **Message Preparation:**

* Convert the plaintext message into a sequence of characters.
* Ensure that the length of the message matches the length of the key.

1. **Encryption:**

* For each character in the message and its corresponding character in the key:
* Convert the character and key character to numeric values (e.g., ASCII or Unicode values).
* Perform a bitwise XOR operation between the two numeric values.
* Convert the result back to a character.
* The encrypted ciphertext is the sequence of characters obtained from the XOR operation.

1. **Decryption:**

* For each character in the ciphertext and its corresponding character in the key:
* Convert the character and key character to numeric values.
* Perform a bitwise XOR operation between the two numeric values.
* Convert the result back to a character.
* The decrypted message is the sequence of characters obtained from the XOR operation.

**PSEUDOCODE:**

import java.security.SecureRandom;

public class One\_Time\_Pad {

public static void main(String[] args) {

String message = "HAYTHAMKENWAY";

// Generate a random key of the same length as the message

String key = generateRandomKey(message.length());

String encryptedMessage = encrypt(message, key);

System.out.println("Message: " + message);

System.out.println("Randomly Generated Key: " + key);

System.out.println("Encrypted Message: " + encryptedMessage);

String decryptedMessage = decrypt(encryptedMessage, key);

System.out.println("Decrypted Message: " + decryptedMessage);

}

public static String generateRandomKey(int length) {

SecureRandom random = new SecureRandom();

StringBuilder key = new StringBuilder();

for (int i = 0; i < length; i++) {

char randomChar = (char) (random.nextInt(26) + 'A');

key.append(randomChar);

}

return key.toString();

}

public static String encrypt(String message, String key) {

StringBuilder encrypted = new StringBuilder();

for (int i = 0; i < message.length(); i++) {

char messageChar = message.charAt(i);

char keyChar = key.charAt(i);

int encryptedChar = (messageChar + keyChar) % 26 + 'A';

encrypted.append((char) encryptedChar);

}

return encrypted.toString();

}

public static String decrypt(String encryptedMessage, String key) {

StringBuilder decrypted = new StringBuilder();

for (int i = 0; i < encryptedMessage.length(); i++) {

char encryptedChar = encryptedMessage.charAt(i);

char keyChar = key.charAt(i);

int decryptedChar = (encryptedChar - keyChar + 26) % 26 + 'A';

decrypted.append((char) decryptedChar);

}

return decrypted.toString();

}

}

**OUTPUT:**

**Message: HAYTHAMKENWAY**

**Randomly Generated Key: IVYKHFBJVCEDR**

**Encrypted Message: PVWDOFNTZPADP**

**Decrypted Message: HAYTHAMKENWAY**

**PRE LAB QUESTIONS**

Here are some pre-lab questions for the given program that computes the objective of the experiment:

1. Describe the basic principles of the Vernam Cipher or One-Time Pad.

2. Why is the One-Time Pad considered unbreakable when used correctly?

3. How does the length of the key in the OTP relate to the length of the plaintext message?

4. What challenges arise in the practical use of the One-Time Pad?

5. How does the OTP ensure both confidentiality and integrity of a message?

**SAMPLE VIVA VOICE QUESTIONS**

Here are some viva voice questions for the given program that computes the objective of the experiment:  
 1. Why is it essential for the key in the One-Time Pad to be truly random?

2. Describe a potential vulnerability of the OTP if the same key is used for multiple messages.

3. In what scenarios would you recommend the use of the OTP despite its practical

challenges?

4. How does the XOR operation play a role in the Vernam Cipher?

5. Can the OTP be used for message authentication? If so, how? If not, why not?

**CONCLUDING REMARKS / GAP ANALYSIS BY THE COURSE FACULTY / COORDINATOR**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

**…………………………………………………………………..**

Faculty name & Signature with date