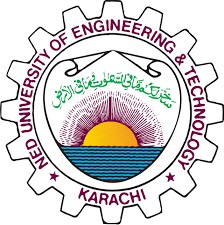
**Object Oriented Programming   
(CT-260)**

Course Project Report



Submitted by:

**Haris Zamir   
Muhammad Raza  
Muhammad Rafay  
Muhammad Nizam**

**CT-125 CT-138  
CT-141  
CT-147**

Program: CSIT Section: C Batch: 2024

Task Distribution:

Muhammad Rafay (CT-141):

* Cipher class
* CipherInterface class
* Ensuring all 5 classes integrate into a single system
* File input
* Design Pattern Implementation

Haris Zamir (CT-125):

* RSA class
* Making of Algorithms (AES, DES, RSA)
* Local mode input
* Code Refinements
* Documentation

Muhammad Nizam (CT-147):

* DES class
* UML Diagram
* Code Testing with Test Cases
* User Interface

Muhammad Raza (CT-138):

* AES class
* UML Diagram
* Error Handling/Debugging
* OOP Implementation review

Title:

***Design a system that encrypts and decrypts files using different encryption algorithms (AES, RSA, DES) with a flexible algorithm selection mechanism.***

Objective:

To encrypt and decrypt files with various different algorithms, each with different specialties and setbacks, and to decide which algorithm the user wants depending on their input. This is done using three different encryption/decryption algorithms which include **AES** (Advanced Encryption Standard), **RSA** (Rivest–Shamir–Adleman) and **DES** (Data Encryption Standard).

Approach:

* We create three different classes for each algorithm, namely **AES**, **RSA** and **DES**.
* We implement algorithmic logic code in each of the three classes to make it functional.
* In all the classes, we will name two protected methods **encrypt** and **decrypt** so that the user be redirected to the algorithm they desire through the parent class.
* We create the **CipherInterface** interface class which contains four pure protected virtual methods which are **encrypt**, **decrypt**, **preprocess** and **postprocess** along with a virtual destructor.
* Inside the **CipherInterface** class, we write the statements for the **encrypt** and **decrypt** method making them follow a pattern which follows preprocess ➡ encrypt/decrypt ➡ postprocess (each of these has a different definition in their respective classes) making it follow the Template design pattern.
* We make **CipherInterface** friends with **Cipher**, so that Cipher can access the private methods in each class.
* We create the **Cipher** class which follows the Strategy design pattern with three public methods **encrypt**, **decrypt** and **setMode** (which takes a CipherInterface pointer as input)
* All classes (AES, DES, RSA) will inherit from the **CipherInterface** abstract class.
* To use them in the main program, we instantiate a **Cipher** object and pass a class object (AES, DES, RSA) into its constructor as reference. This will set that class object as the current strategy.
* To change strategies (cryptography type) we will use the **setMode** method available in the **Cipher** class object.

Design Patterns:

The design patterns used in the above program are:

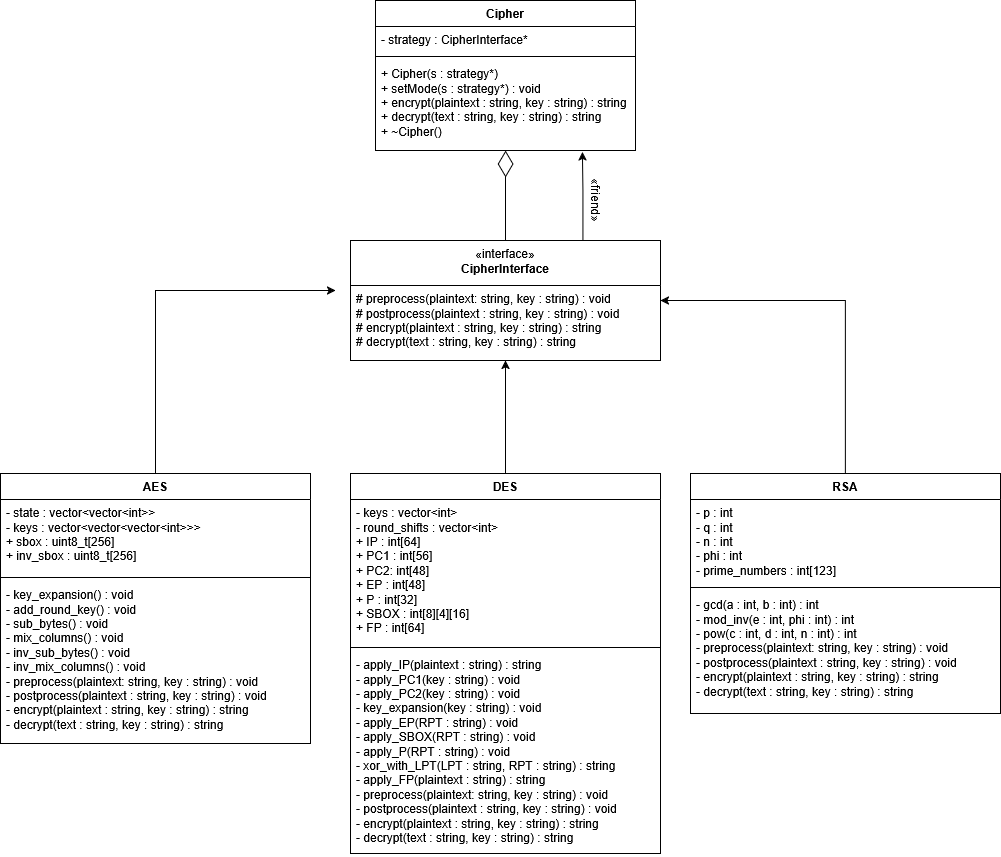
* Strategy Pattern
* Template Method Pattern

Since the **Cipher** class provides a method to change strategies at runtime based on the algorithm the user chooses, it follows the **Strategy Pattern** according to its definition:

*“Defines a family of algorithms and allows them to be interchangeable at runtime.*

Similarly, since the **CipherInterface** class defines the structures of the algorithms (AES, DES, RSA) and executes specific steps with varying results based on the subclasses. It follows the **Template Method Pattern** according to its definition:

*“Defines the structure of an algorithm in a base class but allows subclasses to provide specific implementations of certain steps.”*

UML Diagram:

**How will we encrypt and decrypt files?**

In the *main(int argc, char\* argv[])* function, the user will pass files to the program based on the following CLI format:

*“./cryptography [file1] [file2] ... [fileN] [encrypt|decrypt] [aes|des|rsa]”*

The encryption and decryption will follow the **ECB** (Electronic Codebook) mode of operation where each block is independently encrypted or decrypted based on the algorithms block size.

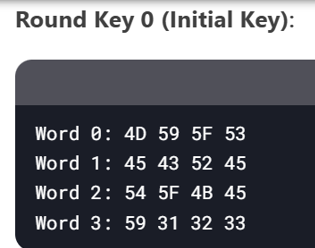
Cryptography Algorithms

Let’s understand the inner working of the three main cryptography algorithms to get a better idea at how they encrypt and decrypt information and what problem they solve.

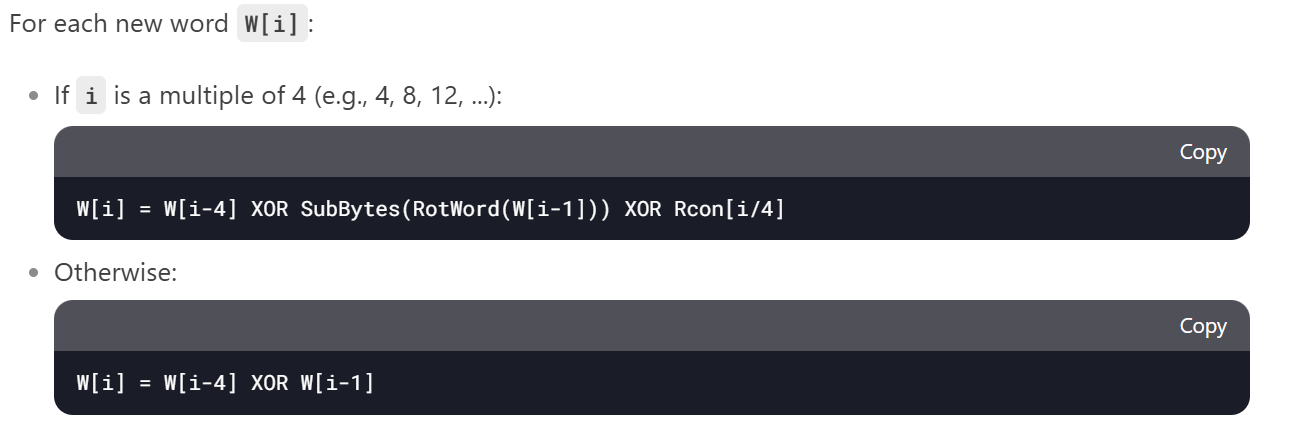
Advanced Encryption Standard *(128-bit)*

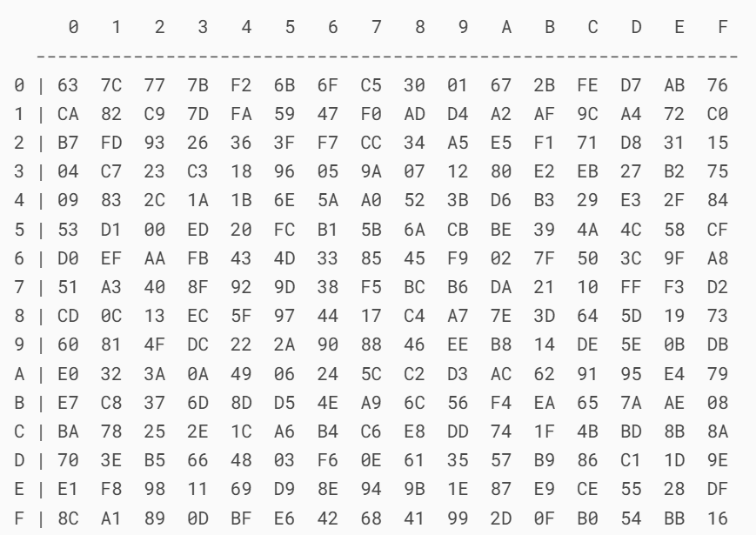
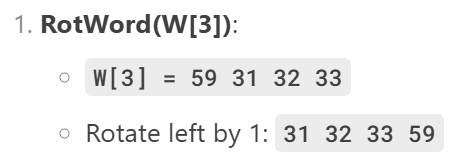
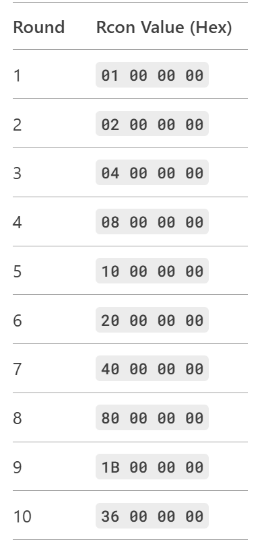
**MAKING OF ROUNDKEYS**

1. Input a 16-byte key from the user and convert it to Hexadecimal. This will be Round Key 0



1. 10 more Round Keys are made to conduct the encryption process, using the formulas below

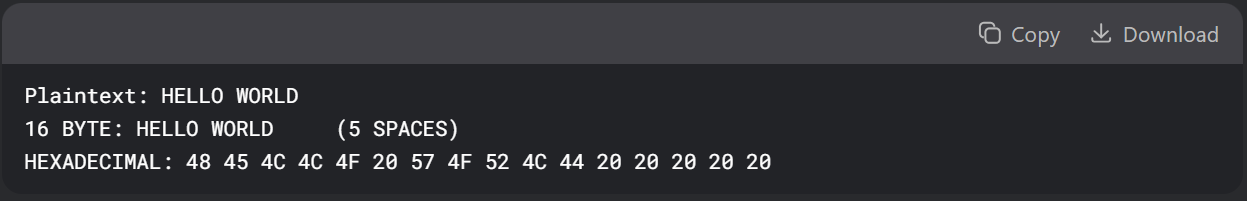


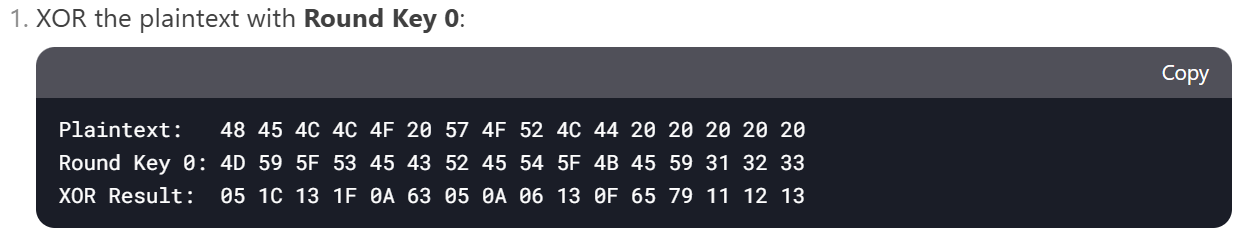


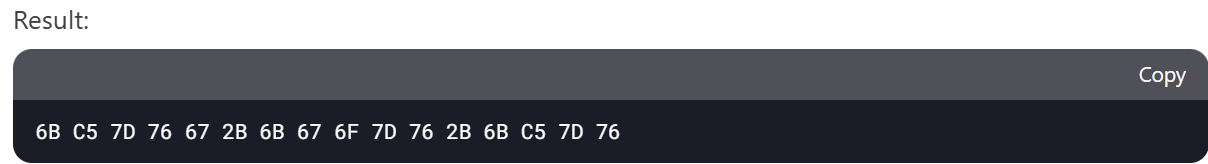
S BOX

**ENCRYPTION**

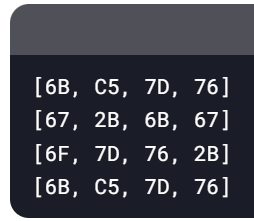
1. Input plain text from the user, convert it to 16 bytes, and convert to hexadecimal



1.  Round 0
2. Round (1-9) (Repeat 9 times)

* Apply S Box on the result of previous round.

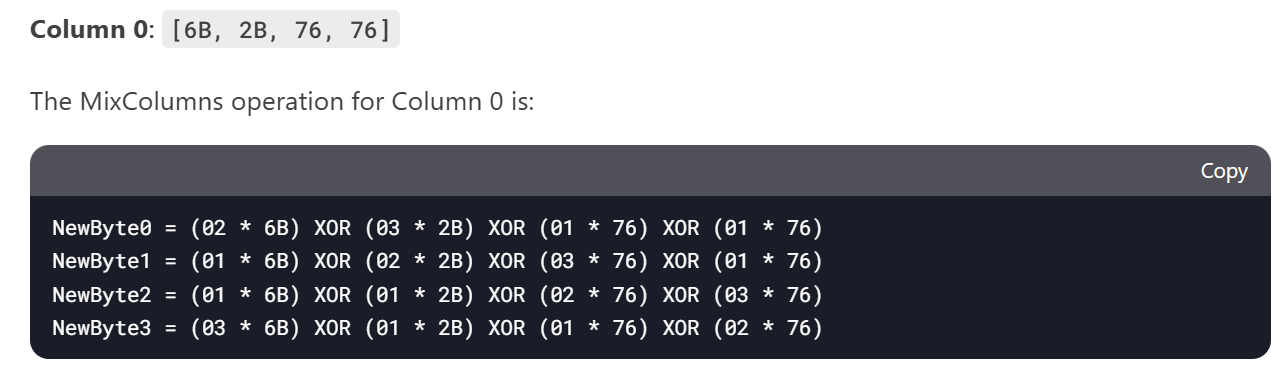
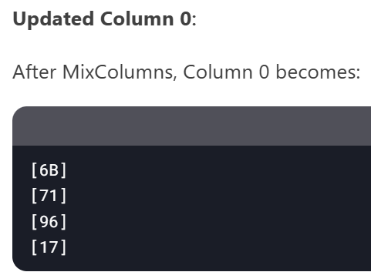
* A screenshot of a computer

  AI-generated content may be incorrect.Shift Rows



* **Row 0**: No shift.
* **Row 1**: Shift left by 1 byte.
* **Row 2**: Shift left by 2 bytes.
* **Row 3**: Shift left by 3 bytes.
* Mix Columns:

0 1 2 3

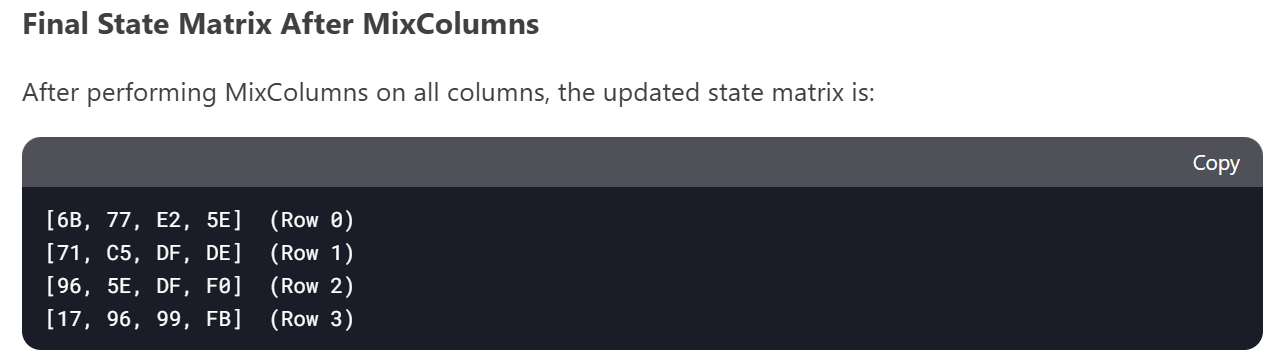


Yellow values differ subject to column (The rest remain same)

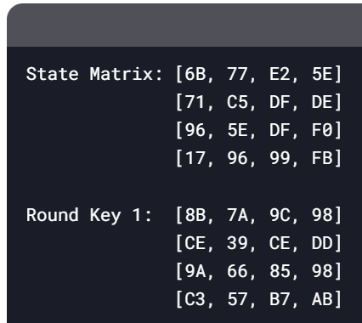
0 1 2 3

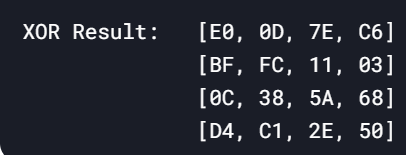


(Repeat for all columns)



* XOR with respective round key (in this case, round key 1)





XOR

1. Round 10

* Repeat Step 3 (Excluding MixColumns)
* The result of Round 10 will be the final encrypted text

**DECRYPTION**

1. Starts with Round key 10 and ends with round key 1
2. A black rectangular object with white letters

   AI-generated content may be incorrect.Round 10
3. Round 9-1 (Repeat 9 times)

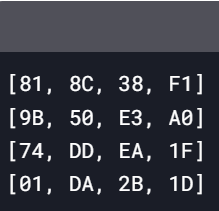
* A black and white screen with white text

  AI-generated content may be incorrect.Inverse Shift Rows

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

* Apply Inverse S Box

A table of numbers and letters

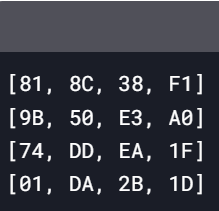
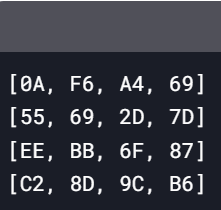
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* XOR with respective Round Key

A black and white screen with white text

AI-generated content may be incorrect.

**Round Key 9 (Example)**

XOR



* Inverse Mix Column

Yellow values differ subject to column (The rest remain same)



0 1 2 3

A screenshot of a computer

AI-generated content may be incorrect.Column 0: [0A, 55, EE, C2]

A black and white screen with white text

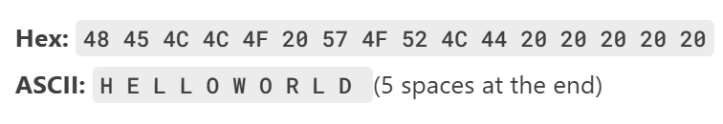
AI-generated content may be incorrect.The Inverse Mix Column operation for Column 0 is

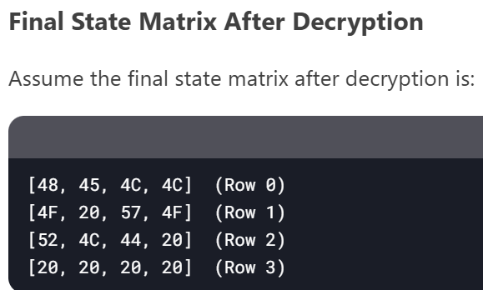
0 1 2 3 3



4. ROUND 0

* Repeat Step 4 (EXCEPT INVERSE MIX COLUMN PART)

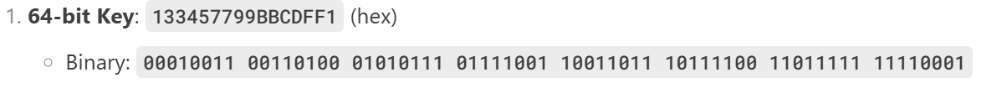
A screenshot of a computer

AI-generated content may be incorrect.  
  
  
  
  
  
 5. Convert Hexadecimal to ASCII to achieve final decrypted text

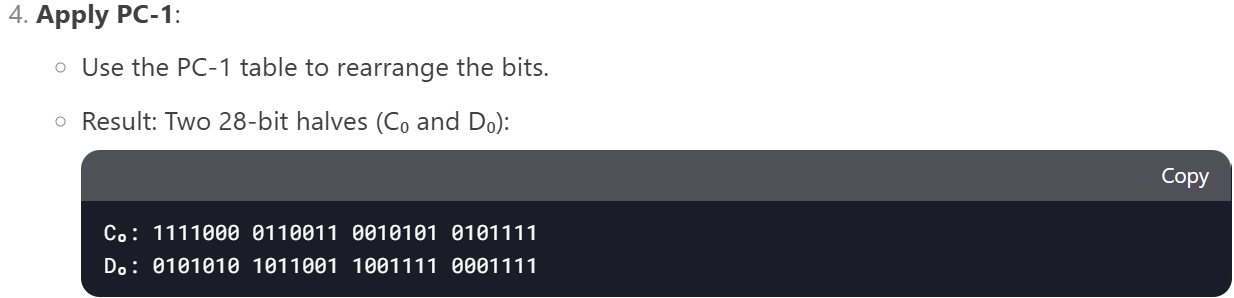
Data Encryption Standard *(56 – bit)*

**MAKING OF SUBKEYS**

1. Make a 16-digit Hexadecimal key (convert to 64-bit binary key)

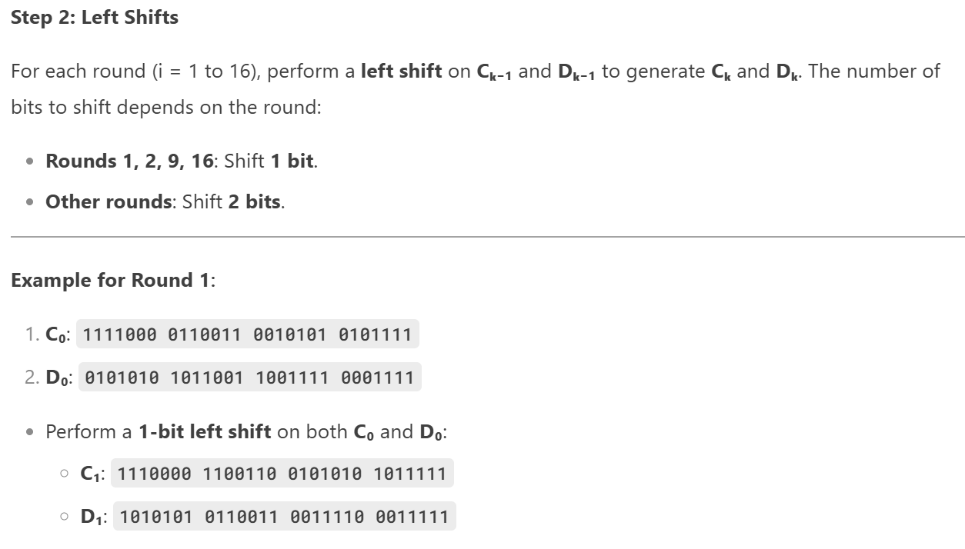


1. A screenshot of a computer

   AI-generated content may be incorrect.Rearrange bits according to PC-1 table, and split in half (This step turns the 64-bit key to a 56-bit key)  
     
    PC-1 TABLE (ORDER OF BITS)

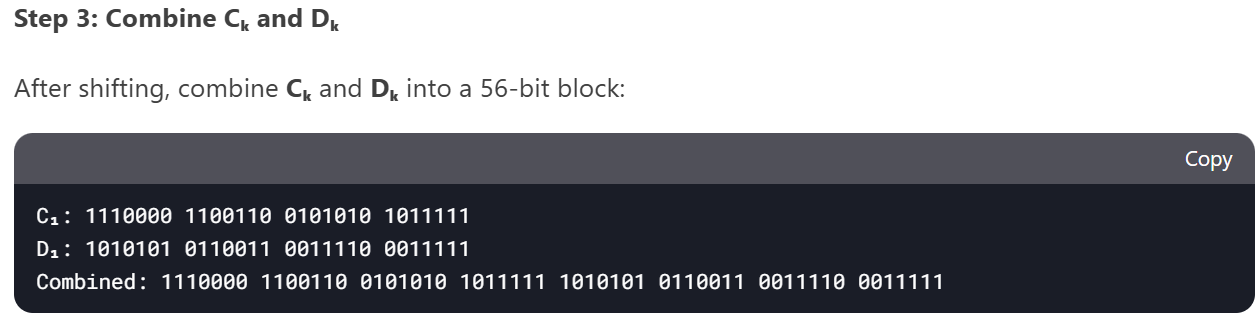
Actually 56

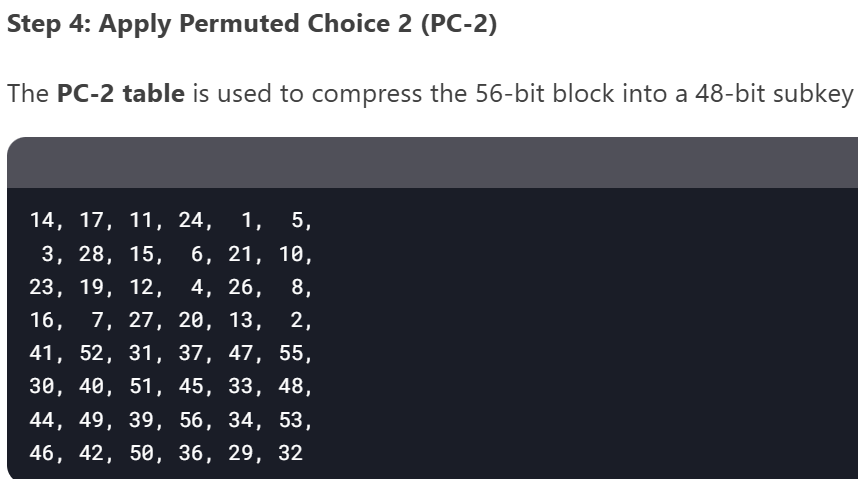
1. Left Shift and PC-2 (repeat 16 times for 16 rounds)

* Perform either 1 or 2 left shifts on the previous 28-bit halves (Ck-1, Dk-1) according to squared data (So for left shift for Round 2, left shift will be applied on C1, D1)

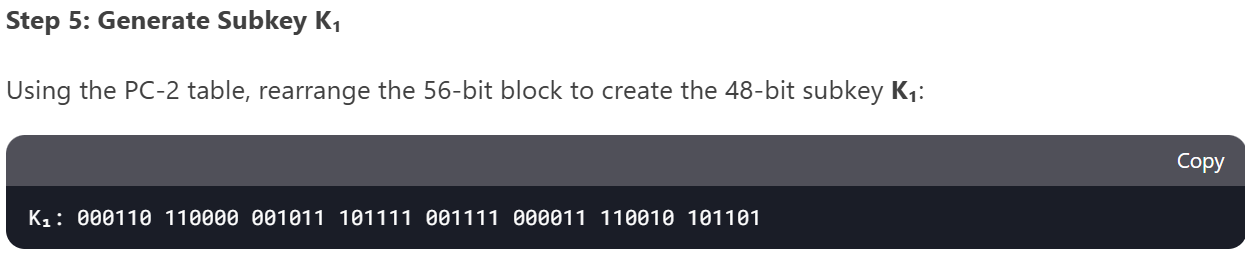


* Combine Ck and Dk



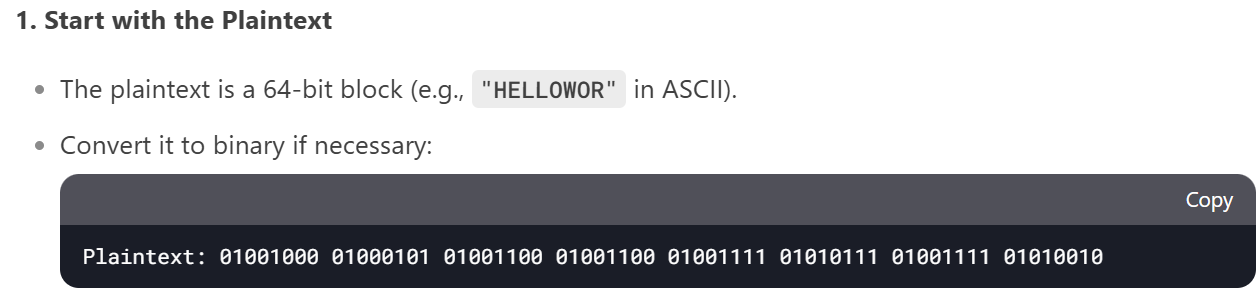
* Rearrange According to PC-2 Table (This step shortens the 56-bit key to 48-bit subkey)

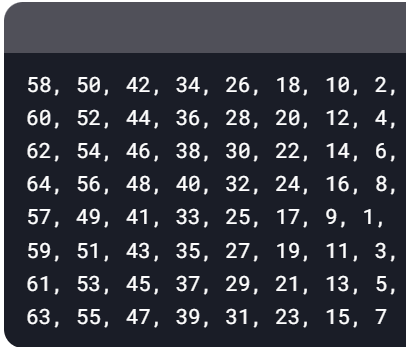
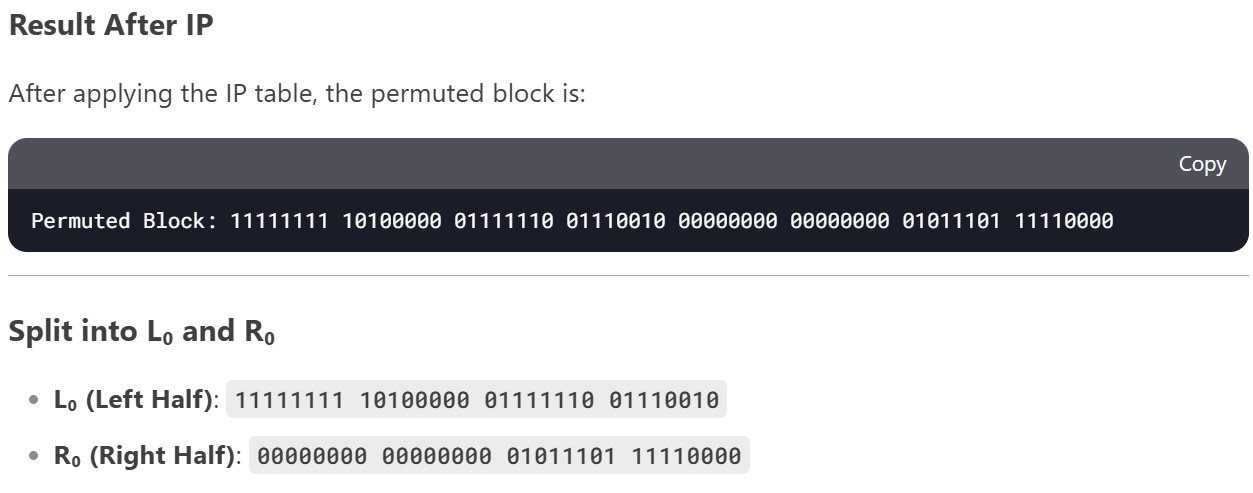
PC-2 TABLE (ORDER OF BITS)



**ENCRYPTION**

1. Convert Chosen 8-character Text to Binary

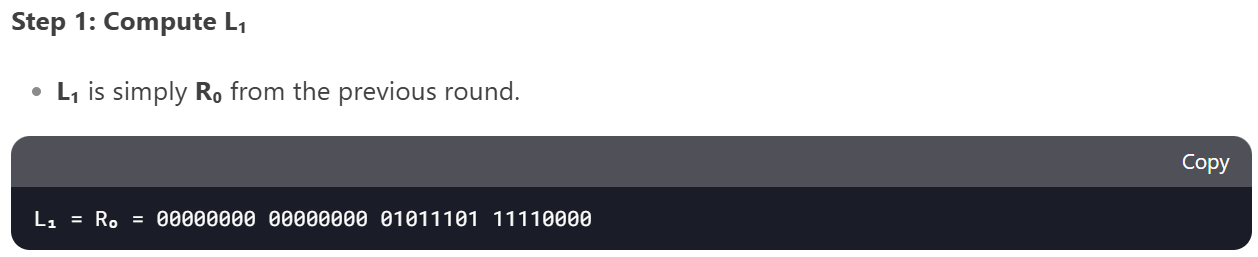


1. Rearrange bits according to IP table, and split in half

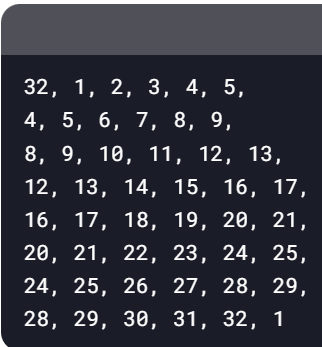
IP TABLE

**Step 3 and 4 are to be repeated 16 times until round 16**

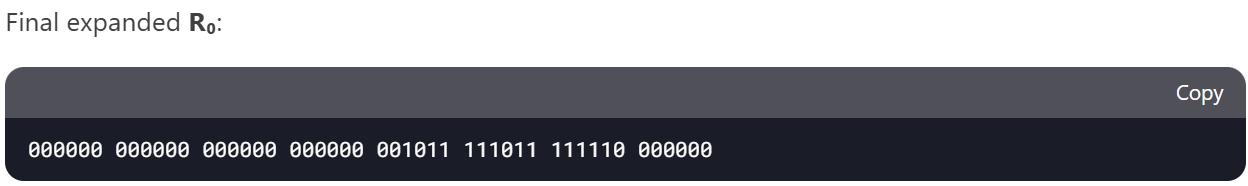
1. Find L1 (ROUND 1)



1. Find R1 (ROUND 1)

* Apply E Table

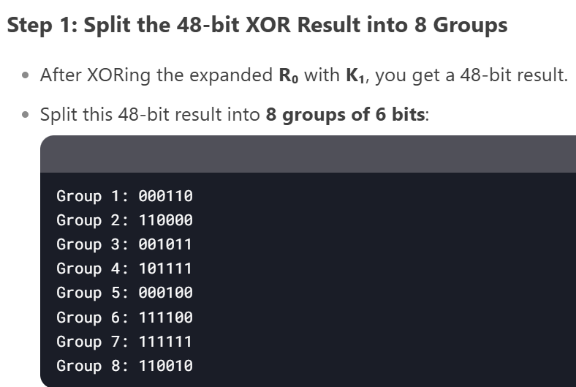
E TABLE (Expands R0)

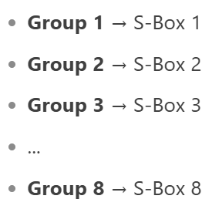


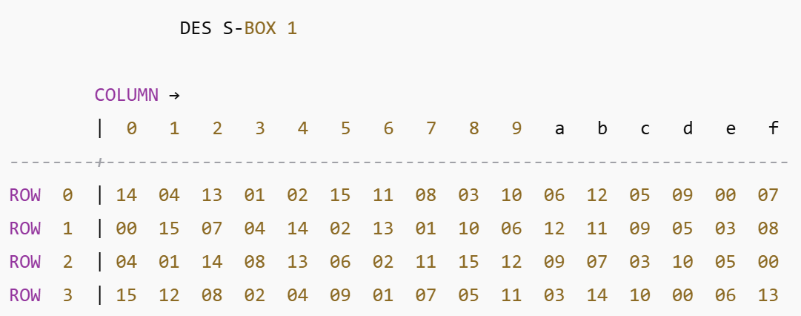
* A black rectangular object with white numbers

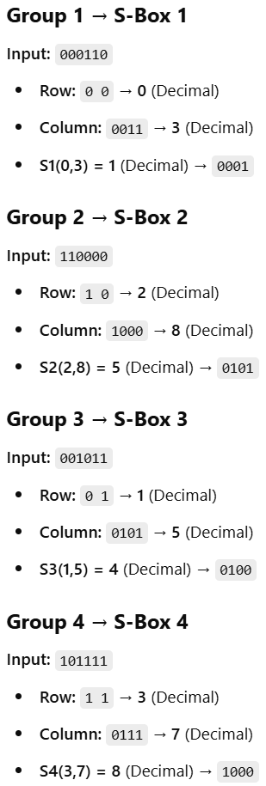
  AI-generated content may be incorrect.XOR with respective Sub-Key
* Apply appropriate S boxes

A white background with black text

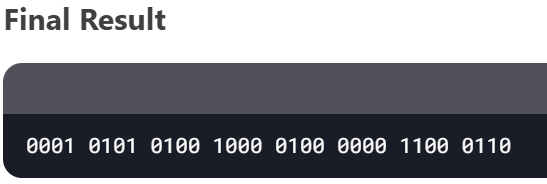
AI-generated content may be incorrect.

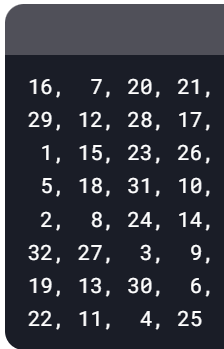


A screenshot of a computer

AI-generated content may be incorrect.

Only S-box 1 included for clarity



* Apply P Table

P TABLE

A close-up of a credit card

AI-generated content may be incorrect. The permutated output is:

* A black and white screen with white text

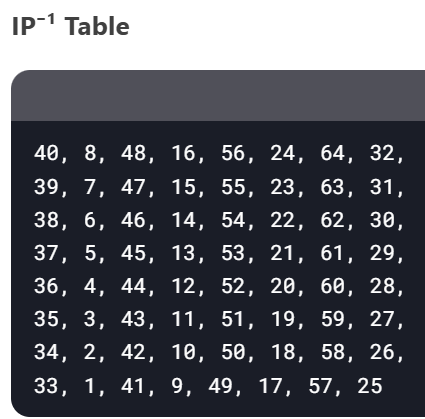
  AI-generated content may be incorrect.XOR With L0 to achieve R1

1. A black rectangular object with a white background

   AI-generated content may be incorrect.Combine L16 and R16

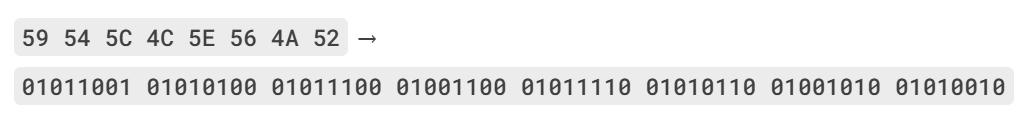
A close-up of a credit card

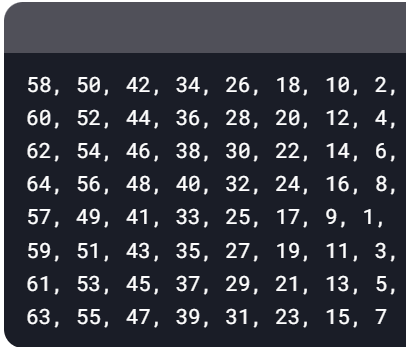
AI-generated content may be incorrect.

* Apply IP⁻¹ Table

**DECRYPTION**

1. Convert cipher text to Binary



1. Rearrange bits according to IP table, and split in half

IP TABLE

A close-up of numbers

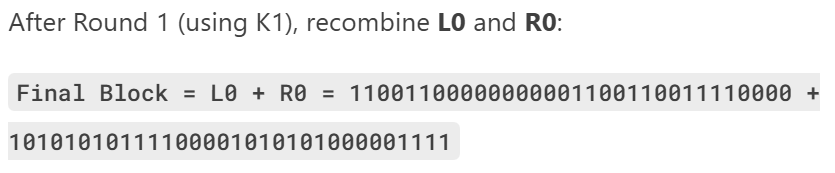
AI-generated content may be incorrect.

1. Compute Lk and Rk till L0 and R0 are achieved (Repeated 16 times)

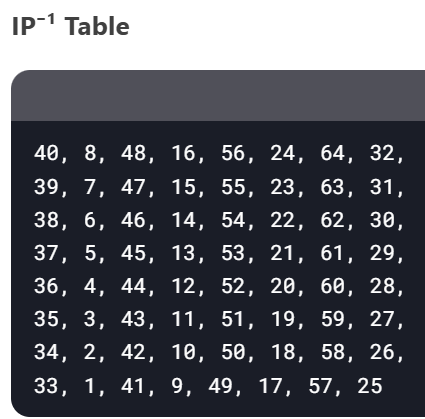
A screenshot of a computer

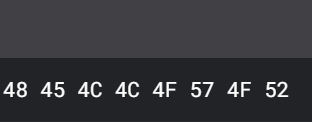
AI-generated content may be incorrect.ROUND 16

1. Combine L0 and R0

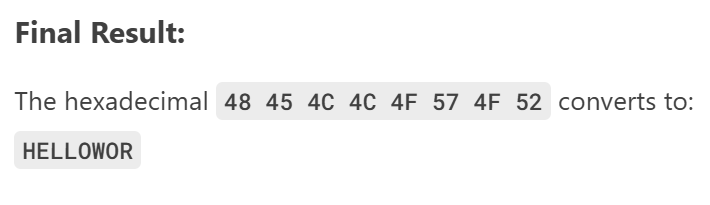


1. Apply IP-1 Table to achieve decrypted text in hexadecimal



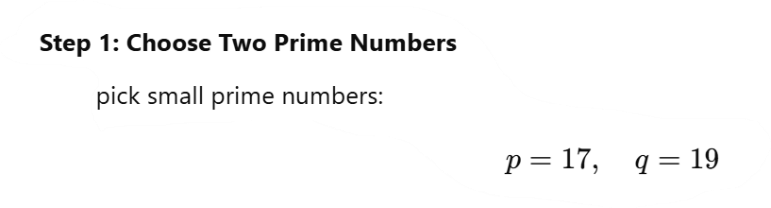


1. Convert the hexadecimal to ASCII to achieve Final Decrypted Text



Rivest–Shamir–Adleman *(9-bit)*

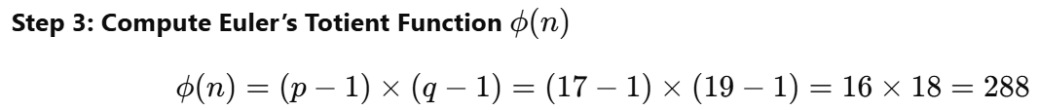
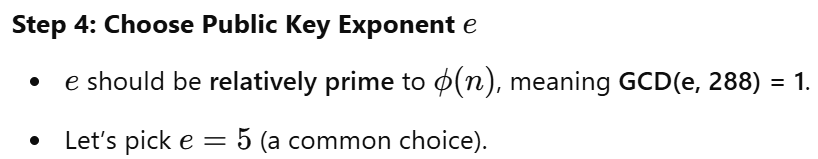
**MAKING OF PUBLIC AND PRIVATE KEYS**

1. Select two prime numbers, p and q

1. Multiply p and q

A white and black background with black text

AI-generated content may be incorrect.

1. Multiply (p-1) with (q-1)
2. Choose e (Public key), where e and ϕ(n) must not have a common factor (other than 1)  
      
    Greatest Common Devisor

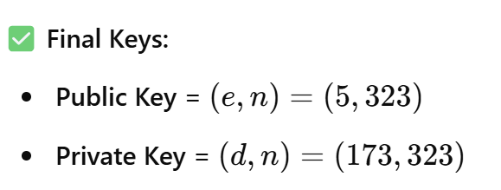
1. Calculate d (Private key), by performing Modular Inverse

A close-up of a sign

AI-generated content may be incorrect.

A black text on a white background

AI-generated content may be incorrect.

1. Final Keys

**ENCRYPTION**

1. Convert text to ASCII

A screenshot of a computer

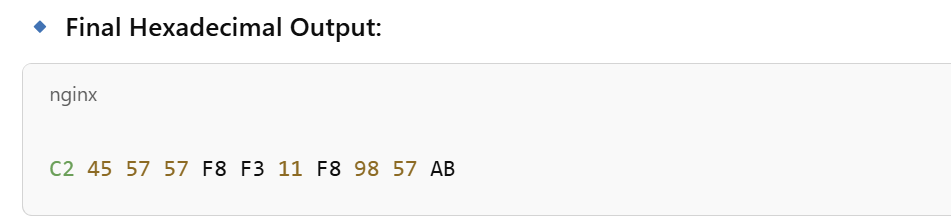
AI-generated content may be incorrect.

1. A white paper with black text and black text

   AI-generated content may be incorrect.Encrypt each letters ASCII value (must be less than n, else make smaller blocks so that it is less than n), using formula and public key

A number and numbers on a white background

AI-generated content may be incorrect.  
  
  
 And rest of the letters...

1. Encrypted Text in Decimal
2.  Convert to Hexadecimal for final Encrypted Text

**DECRYPTION**

1. Reclaim decimal RSA encrypted text

1. Decrypt each number using decryption formula and private key

A white background with black text

AI-generated content may be incorrect.A white background with black text

AI-generated content may be incorrect.

And rest of the blocks...

1. Final Decrypted data

A close-up of a sign

AI-generated content may be incorrect.

Conclusion:

In conclusion, we need a class interface along with a context class (in this case **Cipher**) that will switch between the required strategies (cryptography algorithms) based on the user’s input. And since each encryption and decryption follows a certain number of steps defined in the class interface, the following program includes both the Template and Strategy design pattern.