Name: Shreeyash S. Dongarkar

PRN: 22510025

BTech Final Year CSE

Cryptography and Network Security Lab (B – 1)

Assignment 5 (Apply DES Algorithm for Practical Applications)

**Objectives:**

To implement the DES (Data Encryption Standard) algorithm for encrypting and decrypting a plaintext message to ensure secure data transmission. This lab demonstrates how symmetric key cryptography can be used to protect information in real-world communication systems.

**Problem Statement:**

You are working as a software security analyst for a company that needs to transmit sensitive employee data (like Social Security Numbers or salary information) between two systems over a network. To ensure data confidentiality, you are required to implement the DES algorithm in encryption and decryption mode.   
Your task is to write a program that:

* Takes a plaintext input from the user.
* Uses a user-defined 64-bit key (as a hexadecimal or ASCII string).
* Encrypts the plaintext using the DES algorithm.
* Outputs the encrypted text (ciphertext).
* Decrypts the ciphertext back to plaintext using the same key to verify correctness.

1. import javax.crypto.Cipher;

2. import javax.crypto.SecretKey;

3. import javax.crypto.SecretKeyFactory;

4. import javax.crypto.spec.DESKeySpec;

5. import java.util.Base64;

6. import java.util.Scanner;

7. import java.util.logging.Level;

8. import java.util.logging.Logger;

9.

10. public class DESExample {

11.     private static final Logger logger = Logger.getLogger(DESExample.class.getName());

12.

13.     public static SecretKey generateKey(String keyStr) throws Exception {

14.         byte[] keyBytes = keyStr.getBytes();

15.         if (keyBytes.length != 8) {

16.             throw new IllegalArgumentException("Key must be exactly 8 characters (64 bits).");

17.         }

18.

19.         DESKeySpec desKeySpec = new DESKeySpec(keyBytes);

20.         SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");

21.         return keyFactory.generateSecret(desKeySpec);

22.     }

23.

24.     public static String encrypt(String plaintext, SecretKey key) throws Exception {

25.         Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

26.         cipher.init(Cipher.ENCRYPT\_MODE, key);

27.         byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());

28.         return Base64.getEncoder().encodeToString(encryptedBytes);

29.     }

30.

31.     public static String decrypt(String ciphertext, SecretKey key) throws Exception {

32.         Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

33.         cipher.init(Cipher.DECRYPT\_MODE, key);

34.         byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(ciphertext));

35.         return new String(decryptedBytes);

36.     }

37.

38.     public static void main(String[] args) {

39.         try (Scanner scanner = new Scanner(System.in)) {

40.             System.out.println("==== DES Encryption/Decryption ====");

41.             System.out.println("1. Encrypt");

42.             System.out.println("2. Decrypt");

43.             System.out.print("Enter your choice: ");

44.             int choice = scanner.nextInt();

45.             scanner.nextLine();

46.

47.             System.out.print("Enter 8-character key: ");

48.             String keyStr = scanner.nextLine();

49.             SecretKey key = generateKey(keyStr);

50.

51.             if (choice == 1) {

52.                 System.out.print("Enter plaintext message: ");

53.                 String plaintext = scanner.nextLine();

54.                 String encryptedText = encrypt(plaintext, key);

55.                 System.out.println("Encrypted (Ciphertext): " + encryptedText);

56.             } else if (choice == 2) {

57.                 System.out.print("Enter ciphertext (Base64): ");

58.                 String ciphertext = scanner.nextLine();

59.                 String decryptedText = decrypt(ciphertext, key);

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

61.             } else {

62.                 System.out.println("Invalid choice. Please select 1 or 2.");

63.             }

64.         } catch (IllegalArgumentException e) {

65.             logger.log(Level.WARNING, "Invalid Key: {0}", e.getMessage());

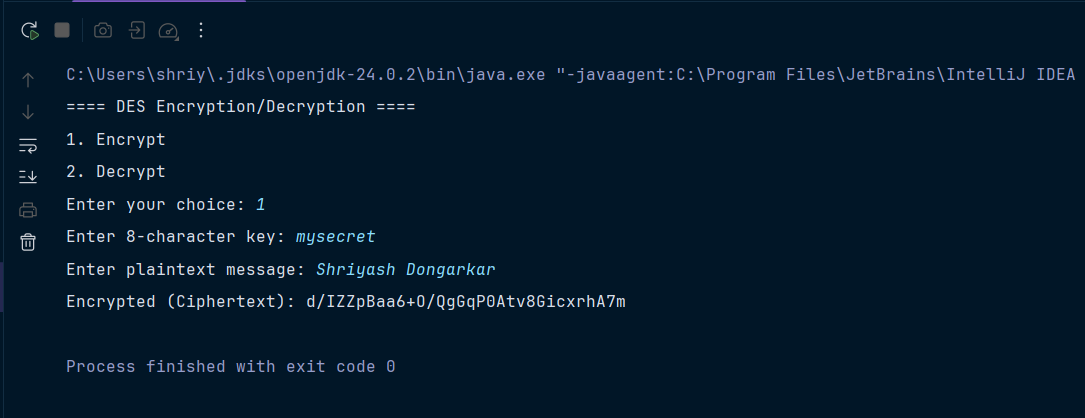
66.         } catch (Exception e) {

67.             logger.log(Level.SEVERE, "An error occurred during DES operation", e);

68.         }

69.     }

70. }





**Tools:**

* **Software:** Java Development Kit (JDK ≥ 8), any IDE (Eclipse, IntelliJ, NetBeans) or text editor
* **Hardware:** Computer system with minimum 4 GB RAM
* **Libraries/Packages:** javax.crypto, java.util.Base64, java.util.Scanner, java.util.logging

**Theory:**

The **Data Encryption Standard (DES)** is a **symmetric key block cipher** developed in the 1970s. It encrypts data in **64-bit blocks** using a **64-bit key** (effectively 56 bits used for security; 8 bits used for parity).

**Key Points:**

* **Block Size:** 64 bits
* **Key Size:** 64 bits (56 effective)
* **Encryption Rounds:** 16 rounds
* **Operations:** Initial Permutation → 16 Feistel rounds (expansion, substitution with S-boxes, permutation) → Final Permutation
* **Symmetric Cipher:** Same key is used for both encryption and decryption.

DES is now considered **insecure** for highly sensitive applications due to its small key size, but it is still studied for educational purposes and forms the basis of more secure algorithms like **Triple DES (3DES)** and **AES**.

**Procedure:**

1. Write a **Java program** that implements DES encryption and decryption using the javax.crypto package.
2. Accept **plaintext input** from the user.
3. Accept an **8-character key** (64-bit) from the user.
4. Encrypt the plaintext using **DES/ECB/PKCS5Padding** mode.
5. Display the **ciphertext (Base64 encoded)**.
6. Decrypt the ciphertext using the same key.
7. Display the **decrypted plaintext** to verify correctness.

**Steps:**

1. Import necessary Java libraries (javax.crypto, java.util.Base64, etc.).
2. Create a method to **generate DES key** from an 8-character string.
3. Create an **encrypt method** using Cipher.getInstance("DES/ECB/PKCS5Padding").
4. Create a **decrypt method** with the same Cipher instance in DECRYPT\_MODE.
5. In the main method:
6. Display a **menu** (1 → Encrypt, 2 → Decrypt).
7. Accept user input for plaintext/ciphertext and key.
8. Perform the selected operation.
9. Print results.
10. Run the program and record the output.

**Observations:**

* Plaintext is successfully converted to unreadable ciphertext using DES.
* Decrypting the ciphertext with the **same key** restores the original plaintext.
* Changing even **1 character in the key** results in a completely different ciphertext and decryption failure.
* Output ciphertext differs for different plaintext messages, showing DES’s **diffusion property**.

**Conclusion:**

* The DES algorithm was successfully implemented in Java for both encryption and decryption.
* It demonstrates how **symmetric key cryptography** ensures **data confidentiality** in communication.
* While DES is now outdated for modern secure applications (due to brute-force vulnerability), the experiment highlights the **principles of block ciphers and symmetric encryption**, serving as a foundation for advanced algorithms like **AES**.