Name: Shreeyash S. Dongarkar

PRN: 22510025

BTech Final Year CSE

Cryptography and Network Security Lab (B – 1)

Assignment 2: Encryption and Decryption Using Transposition Ciphers

**Objective**:

* To understand and implement encryption and decryption using the Rail Fence cipher.
* To understand and implement encryption and decryption using the Row and Column Transposition cipher.

1) Rail Fence Cipher

1. import java.util.Scanner;

2.

3. public class RailFenceCipher {

4.

5.     private static String encrypt(String text, int rails) {

6.         text = text.replaceAll("\\s+", "");

7.         char[][] rail = new char[rails][text.length()];

8.

9.         for (int i = 0; i < rails; i++)

10.             for (int j = 0; j < text.length(); j++)

11.                 rail[i][j] = '\n';

12.

13.         boolean dirDown = false;

14.         int row = 0, col = 0;

15.

16.         for (char c : text.toCharArray()) {

17.             if (row == 0 || row == rails - 1) dirDown = !dirDown;

18.             rail[row][col++] = c;

19.             row += dirDown ? 1 : -1;

20.         }

21.

22.         StringBuilder result = new StringBuilder();

23.         for (int i = 0; i < rails; i++)

24.             for (int j = 0; j < text.length(); j++)

25.                 if (rail[i][j] != '\n')

26.                     result.append(rail[i][j]);

27.

28.         return result.toString();

29.     }

30.

31.     private static String decrypt(String cipher, int rails) {

32.         char[][] rail = new char[rails][cipher.length()];

33.

34.         for (int i = 0; i < rails; i++)

35.             for (int j = 0; j < cipher.length(); j++)

36.                 rail[i][j] = '\n';

37.

38.         boolean dirDown = false;

39.         int row = 0, col = 0;

40.

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

42.             if (row == 0 || row == rails - 1) dirDown = !dirDown;

43.             rail[row][col++] = '\*';

44.             row += dirDown ? 1 : -1;

45.         }

46.

47.         int index = 0;

48.         for (int i = 0; i < rails; i++) {

49.             for (int j = 0; j < cipher.length(); j++) {

50.                 if (rail[i][j] == '\*' && index < cipher.length()) {

51.                     rail[i][j] = cipher.charAt(index++);

52.                 }

53.             }

54.         }

55.

56.         StringBuilder result = new StringBuilder();

57.         row = 0; col = 0;

58.         dirDown = false;

59.

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

61.             if (row == 0 || row == rails - 1) dirDown = !dirDown;

62.             result.append(rail[row][col++]);

63.             row += dirDown ? 1 : -1;

64.         }

65.

66.         return result.toString();

67.     }

68.

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

70.         Scanner sc = new Scanner(System.in);

71.

72.         System.out.println("Rail Fence Cipher");

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

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

75.         System.out.print("Choose an option (1 or 2): ");

76.         int choice = sc.nextInt();

77.         sc.nextLine();

78.

79.         System.out.print("Enter the number of rails: ");

80.         int rails = sc.nextInt();

81.         sc.nextLine();

82.

83.         System.out.print("Enter the text: ");

84.         String text = sc.nextLine();

85.

86.         String result;

87.         if (choice == 1) {

88.             result = encrypt(text, rails);

89.             System.out.println("Encrypted Text: " + result);

90.         } else if (choice == 2) {

91.             result = decrypt(text, rails);

92.             System.out.println("Decrypted Text: " + result);

93.         } else {

94.             System.out.println("Invalid choice.");

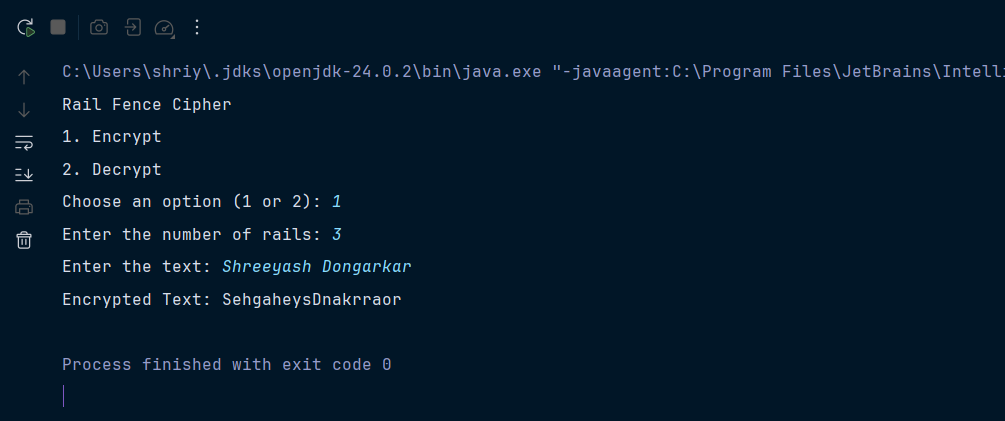
95.         }

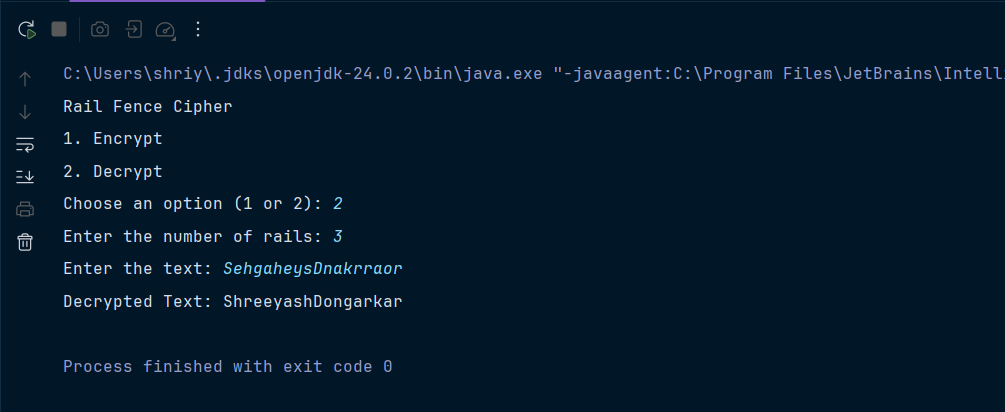
96.

97.         sc.close();

98.     }

99. }





2) Row Column Transposition Cipher

1. import java.util.\*;

2.

3. public class RowColumnTranspositionCipher {

4.

5.     public static String encrypt(String plaintext, String key) {

6.         plaintext = plaintext.replaceAll("\\s", "").toUpperCase();

7.         int cols = key.length();

8.         int rows = (int) Math.ceil((double) plaintext.length() / cols);

9.

10.         char[][] matrix = new char[rows][cols];

11.         int k = 0;

12.         for (int i = 0; i < rows; i++) {

13.             for (int j = 0; j < cols; j++) {

14.                 if (k < plaintext.length())

15.                     matrix[i][j] = plaintext.charAt(k++);

16.                 else

17.                     matrix[i][j] = 'X';

18.             }

19.         }

20.

21.         Integer[] order = getOrder(key);

22.

23.         StringBuilder cipher = new StringBuilder();

24.         for (int col : order) {

25.             for (int i = 0; i < rows; i++) {

26.                 cipher.append(matrix[i][col]);

27.             }

28.         }

29.         return cipher.toString();

30.     }

31.

32.     public static String decrypt(String ciphertext, String key) {

33.         int cols = key.length();

34.         int rows = (int) Math.ceil((double) ciphertext.length() / cols);

35.

36.         char[][] matrix = new char[rows][cols];

37.

38.         Integer[] order = getOrder(key);

39.

40.         int k = 0;

41.         for (int col : order) {

42.             for (int i = 0; i < rows; i++) {

43.                 if (k < ciphertext.length())

44.                     matrix[i][col] = ciphertext.charAt(k++);

45.             }

46.         }

47.

48.         StringBuilder plain = new StringBuilder();

49.         for (int i = 0; i < rows; i++) {

50.             for (int j = 0; j < cols; j++) {

51.                 plain.append(matrix[i][j]);

52.             }

53.         }

54.         return plain.toString().replace("X", "");

55.     }

56.

57.     private static Integer[] getOrder(String key) {

58.         key = key.toUpperCase();

59.         Character[] keyChars = new Character[key.length()];

60.         for (int i = 0; i < key.length(); i++) keyChars[i] = key.charAt(i);

61.

62.         Integer[] order = new Integer[key.length()];

63.         for (int i = 0; i < key.length(); i++) order[i] = i;

64.

65.         Arrays.sort(order, (a, b) -> keyChars[a].compareTo(keyChars[b]));

66.         return order;

67.     }

68.

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

70.         Scanner sc = new Scanner(System.in);

71.

72.         System.out.println("Row and Column Transposition Cipher");

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

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

75.         System.out.print("Choose an option (1 or 2): ");

76.         int choice = sc.nextInt();

77.         sc.nextLine();

78.

79.         System.out.print("Enter the key: ");

80.         String key = sc.nextLine();

81.

82.         System.out.print("Enter the text: ");

83.         String text = sc.nextLine();

84.

85.         String result;

86.         if (choice == 1) {

87.             result = encrypt(text, key);

88.             System.out.println("Encrypted Text: " + result);

89.         } else if (choice == 2) {

90.             result = decrypt(text, key);

91.             System.out.println("Decrypted Text: " + result);

92.         } else {

93.             System.out.println("Invalid choice.");

94.         }

95.

96.         sc.close();

97.     }

98. }

