**Cryptography and Network Security Lab Manual**

**1.Write a Java 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.**

**Program:**

public class XORWithZero

{

public static void main(String[] args)

{

// Define the string

String text = "Hello World";

// Display the original string

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

// Perform XOR operation with 0 and display the result System.out.print("XOR with 0: ");

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

{

char c = text.charAt(i);

char xorResult = (char)(c ^ 0);

// XOR with 0 System.out.print(xorResult);

}

System.out.println();

}

}

**Output:**

Original String: Hello World

XOR with 0: Hello World

**2.Write a java 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 displays the result.**

**Program:**

public class BitwiseOperations

{

public static void main(String[] args)

{

// Define the string

String text = "Hello World";

// Display the original string

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

// Perform AND operation with 127 and display the result System.out.print("AND with 127: ");

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

{

char c = text.charAt(i);

char andResult = (char)(c & 127);

System.out.print(andResult);

}

System.out.println();

// Perform XOR operation with 127 and display the result System.out.print("XOR with 127: ");

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

{

char c = text.charAt(i);

char xorResult = (char)(c ^ 127);

System.out.print(xorResult);

}

System.out.println();

}

}

**Output:**

Original String: Hello World

AND with 127: Hello World XOR with 127: 7xqq~?pq~q

**3. A)Write a java program to perform encryption and decryption using the Ceaser cipher algorithm?**

**Program:**

import java.util.Scanner;

public class CaesarCipher {

// Method to encrypt the message using Caesar Cipher

public static String encrypt(String message, int shift) {

StringBuilder result = new StringBuilder();

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

char ch = message.charAt(i);

// Encrypt uppercase letters

if (Character.isUpperCase(ch)) {

char c = (char) (((int) ch + shift - 65) % 26 + 65);

result.append(c);

}

// Encrypt lowercase letters

else if (Character.isLowerCase(ch)) {

char c = (char) (((int) ch + shift - 97) % 26 + 97);

result.append(c);

}

// Keep non-alphabetic characters as they are

else {

result.append(ch);

}

}

return result.toString();

}

// Method to decrypt the message using Caesar Cipher

public static String decrypt(String message, int shift) {

return encrypt(message, 26 - shift); // Decrypt is reverse of encrypt with 26 - shift

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the message and shift value

System.out.print("Enter the message: ");

String message = scanner.nextLine();

System.out.print("Enter the shift value (1-25): ");

int shift = scanner.nextInt();

// Input validation for shift value

if (shift < 1 || shift > 25) {

System.out.println("Invalid shift value. Please enter a number between 1 and 25.");

return;

}

// Encrypt the message

String encryptedMessage = encrypt(message, shift);

System.out.println("Encrypted Message: " + encryptedMessage);

// Decrypt the message

String decryptedMessage = decrypt(encryptedMessage, shift);

System.out.println("Decrypted Message: " + decryptedMessage);

scanner.close();

}

}

### Output:

Enter the message: Hello World

Enter the shift value (1-25): 1

Encrypted Message: Ifmmp Xpsme

Decrypted Message: Hello World

=== Code Execution Successful ===

**3.B)Write a java program to perform encryption and decryption using the Substitution cipher algorithm?**

**Program:**

import java.util.Scanner;

public class SubstitutionCipher {

// Alphabet used for reference

private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

// Method to encrypt the message using Substitution Cipher

public static String encrypt(String message, String key) {

StringBuilder encryptedMessage = new StringBuilder();

message = message.toUpperCase();

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

char currentChar = message.charAt(i);

// If character is an alphabetic letter

if (Character.isLetter(currentChar)) {

int indexInAlphabet = ALPHABET.indexOf(currentChar);

char encryptedChar = key.charAt(indexInAlphabet);

encryptedMessage.append(encryptedChar);

} else {

// Non-alphabet characters are added as-is

encryptedMessage.append(currentChar);

}

}

return encryptedMessage.toString();

}

// Method to decrypt the message using Substitution Cipher

public static String decrypt(String encryptedMessage, String key) {

StringBuilder decryptedMessage = new StringBuilder();

encryptedMessage = encryptedMessage.toUpperCase();

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

char currentChar = encryptedMessage.charAt(i);

// If character is an alphabetic letter

if (Character.isLetter(currentChar)) {

int indexInKey = key.indexOf(currentChar);

char decryptedChar = ALPHABET.charAt(indexInKey);

decryptedMessage.append(decryptedChar);

} else {

// Non-alphabet characters are added as-is

decryptedMessage.append(currentChar);

}

}

return decryptedMessage.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Define the substitution key (26 unique uppercase letters)

String key = "QWERTYUIOPLKJHGFDSAZXCVBNM"; // Example key, can be any permutation of 26 letters

System.out.println("Using substitution key: " + key);

// Input the message to encrypt

System.out.print("Enter the message to encrypt: ");

String message = scanner.nextLine();

// Encrypt the message

String encryptedMessage = encrypt(message, key);

System.out.println("Encrypted Message: " + encryptedMessage);

// Decrypt the message

String decryptedMessage = decrypt(encryptedMessage, key);

System.out.println("Decrypted Message: " + decryptedMessage);

scanner.close();

}

}

**Output:**

Using substitution key: QWERTYUIOPLKJHGFDSAZXCVBNM

Enter the message to encrypt: RAMA

Encrypted Message: SQJQ

Decrypted Message: RAMA

**3.C)Write a java program to perform encryption and decryption using the Hill cipher algorithm?**

**Program:**

import java.util.Scanner;

public class HillCipher

{

// Function to perform matrix multiplication

public static int[] matrixMultiply(int[][] keyMatrix, int[] messageVector) {

int[] result = new int[messageVector.length];

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

result[i] = 0;

for (int j = 0; j < keyMatrix[i].length; j++) {

result[i] += keyMatrix[i][j] \* messageVector[j];

}

result[i] = result[i] % 26; // Perform modulo 26 operation

}

return result;

}

// Function to find the modular inverse of a number

public static int modInverse(int a, int m) {

a = a % m;

for (int x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return 1;

}

// Function to calculate the inverse of a 2x2 matrix

public static int[][] inverseKeyMatrix(int[][] keyMatrix) {

int determinant = (keyMatrix[0][0] \* keyMatrix[1][1] - keyMatrix[0][1] \* keyMatrix[1][0]) % 26;

determinant = (determinant + 26) % 26;

int inverseDeterminant = modInverse(determinant, 26);

int[][] inverseMatrix = new int[2][2];

inverseMatrix[0][0] = (keyMatrix[1][1] \* inverseDeterminant) % 26;

inverseMatrix[1][1] = (keyMatrix[0][0] \* inverseDeterminant) % 26;

inverseMatrix[0][1] = (-keyMatrix[0][1] \* inverseDeterminant + 26) % 26;

inverseMatrix[1][0] = (-keyMatrix[1][0] \* inverseDeterminant + 26) % 26;

return inverseMatrix;

}

// Function to convert a string into an integer vector

public static int[] stringToVector(String text) {

int[] vector = new int[text.length()];

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

vector[i] = text.charAt(i) - 'A';

}

return vector;

}

// Function to convert an integer vector into a string

public static String vectorToString(int[] vector) {

StringBuilder text = new StringBuilder();

for (int i : vector) {

text.append((char) (i + 'A'));

}

return text.toString();

}

// Function to encrypt the plaintext

public static String encrypt(String plaintext, int[][] keyMatrix) {

int[] messageVector = stringToVector(plaintext);

int[] encryptedVector = matrixMultiply(keyMatrix, messageVector);

return vectorToString(encryptedVector);

}

// Function to decrypt the ciphertext

public static String decrypt(String ciphertext, int[][] keyMatrix) {

int[][] inverseMatrix = inverseKeyMatrix(keyMatrix);

int[] messageVector = stringToVector(ciphertext);

int[] decryptedVector = matrixMultiply(inverseMatrix, messageVector);

return vectorToString(decryptedVector);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input: 2x2 key matrix

int[][] keyMatrix = new int[2][2];

System.out.println("Enter the 2x2 key matrix (values between 0 and 25):");

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

for (int j = 0; j < 2; j++) {

keyMatrix[i][j] = scanner.nextInt();

}

}

// Input: plaintext (must be of length 2 for simplicity)

System.out.println("Enter the plaintext (length 2, uppercase letters only):");

String plaintext = scanner.next().toUpperCase();

// Encrypt the plaintext

String ciphertext = encrypt(plaintext, keyMatrix);

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

// Decrypt the ciphertext

String decryptedText = decrypt(ciphertext, keyMatrix);

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

scanner.close();

}

}

Output:

1 2

3 4

Enter the plaintext (length 2, uppercase letters only):

AB

Encrypted Text: CE

Decrypted Text: AY

=== Code Execution Successful ===

**4.Write a java program to implement the DES algorithm logic?**

**Program:**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class DESExample {

// Method to generate a secret key for DES

public static SecretKey generateKey() throws Exception {

KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");

keyGenerator.init(56); // DES uses a 56-bit key size

return keyGenerator.generateKey();

}

// Method to encrypt data using the DES algorithm

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

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

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

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

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

}

// Method to decrypt data using the DES algorithm

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

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

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

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

return new String(decryptedBytes);

}

public static void main(String[] args) {

try {

// Generate a secret key for DES

SecretKey secretKey = generateKey();

// Plain text to be encrypted

String plaintext = "Hello, World!";

System.out.println("Original Text: " + plaintext);

// Encrypt the plain text

String encryptedText = encrypt(plaintext, secretKey);

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

// Decrypt the encrypted text

String decryptedText = decrypt(encryptedText, secretKey);

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

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Output:**

Original Text: Hello, World!

Encrypted Text: wEm+7nd6ij+aOwmOMdQORQ==

Decrypted Text: Hello, World!

**5.Write a java program to implement the Blowfish algorithm logic?**

**Program:**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class BlowfishExample {

// Method to generate a secret key for Blowfish

public static SecretKey generateKey(int keySize) throws Exception {

KeyGenerator keyGenerator = KeyGenerator.getInstance("Blowfish");

keyGenerator.init(keySize); // keySize can be between 32 and 448 bits

return keyGenerator.generateKey();

}

// Method to encrypt data using the Blowfish algorithm

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

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

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

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

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

}

// Method to decrypt data using the Blowfish algorithm

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

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

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

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

return new String(decryptedBytes);

}

public static void main(String[] args) {

try {

// Generate a secret key for Blowfish

SecretKey secretKey = generateKey(128); // You can specify a key size between 32 and 448 bits

// Plain text to be encrypted

String plaintext = "Hello, World!";

System.out.println("Original Text: " + plaintext);

// Encrypt the plain text

String encryptedText = encrypt(plaintext, secretKey);

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

// Decrypt the encrypted text

String decryptedText = decrypt(encryptedText, secretKey);

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

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Out Put:**

* Original Text: Hello, World!
* Encrypted Text: XNcjWiCOqfEnr6Fjc8GViw==
* Decrypted Text: Hello, World!
* === Code Execution Successful ===

6.W**rite a java program to implement the Rijndael algorithm logic?**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class AESExample {

// Method to generate a secret key

public static SecretKey generateKey(int n) throws Exception {

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

keyGenerator.init(n);

return keyGenerator.generateKey();

}

// Method to encrypt data using the AES algorithm

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

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

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

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

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

}

// Method to decrypt data using the AES algorithm

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

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

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

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

return new String(decryptedBytes);

}

public static void main(String[] args) {

try {

// Generate a secret key for AES

SecretKey secretKey = generateKey(128);

// Plain text to be encrypted

String plaintext = "Hello, World!";

System.out.println("Original Text: " + plaintext);

// Encrypt the plain text

String encryptedText = encrypt(plaintext, secretKey);

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

// Decrypt the encrypted text

String decryptedText = decrypt(encryptedText, secretKey);

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

} catch (Exception e) {

e.printStackTrace();

}

}

}

**OutPut:**

Original Text: Hello, World!

Encrypted Text: AYss0loz6Ml+kWPZ8lj6bA==

Decrypted Text: Hello, World!

**=== Code Execution Successful ===**

**7.Write a java program the RC4 logic using cryptography; encrypt the text "Hello World" using Blowfish. Create your own key using java key tool?**

**Program**:

import java.util.Scanner;

public class RC4 {

private byte[] S = new byte[256];

private int x = 0;

private int y = 0;

// Constructor to initialize the key

public RC4(byte[] key) {

init(key);

}

// Initialize the permutation in the array S

private void init(byte[] key) {

int keyLength = key.length;

for (int i = 0; i < 256; i++) {

S[i] = (byte) i;

}

int j = 0;

for (int i = 0; i < 256; i++) {

j = (j + S[i] + key[i % keyLength]) & 0xFF;

swap(i, j);

}

}

// Swap elements in the array S

private void swap(int i, int j) {

byte temp = S[i];

S[i] = S[j];

S[j] = temp;

}

// Generate the key stream and perform encryption/decryption

public byte[] encrypt(byte[] plaintext) {

byte[] ciphertext = new byte[plaintext.length];

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

ciphertext[i] = (byte) (plaintext[i] ^ keyItem());

}

return ciphertext;

}

// Generate the next byte of the key stream

private byte keyItem() {

x = (x + 1) & 0xFF;

y = (y + S[x]) & 0xFF;

swap(x, y);

return S[(S[x] + S[y]) & 0xFF];

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a key for RC4 encryption (e.g., mysecretkey):");

String keyString = scanner.nextLine();

byte[] key = keyString.getBytes();

RC4 rc4 = new RC4(key);

String plaintext = "Hello World";

System.out.println("Original Text: " + plaintext);

byte[] ciphertext = rc4.encrypt(plaintext.getBytes());

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

// Decrypting the ciphertext

byte[] decryptedText = rc4.encrypt(ciphertext); // RC4 is symmetric, so encryption and decryption are the same

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

scanner.close();

}

}

**Output:**

Enter a key for RC4 encryption (e.g., mysecretkey):

1

Original Text: Hello World

Encrypted Text: (??g?\_x001D\_P

Decrypted Text:?5??.1LW.?V

=== Code Execution Successful ===

8.W**rite a java program to implement RSA algorithm?**

**Program:**

import java.math.BigInteger;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.spec.RSAPrivateKeySpec;

import java.security.spec.RSAPublicKeySpec;

import javax.crypto.Cipher;

public class RSAExample {

public static void main(String[] args) {

try {

// Generate RSA key pair

KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");

keyPairGenerator.initialize(2048); // Key size (2048 bits for strong security)

KeyPair keyPair = keyPairGenerator.generateKeyPair();

PublicKey publicKey = keyPair.getPublic();

PrivateKey privateKey = keyPair.getPrivate();

// Print the key details

printKeyDetails(publicKey, privateKey);

// Text to be encrypted

String plaintext = "Hello, RSA!";

System.out.println("Original Text: " + plaintext);

// Encrypt the text using the public key

byte[] encryptedText = encrypt(plaintext, publicKey);

System.out.println("Encrypted Text: " + new String(encryptedText));

// Decrypt the text using the private key

String decryptedText = decrypt(encryptedText, privateKey);

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

} catch (Exception e) {

e.printStackTrace();

}

}

// Method to encrypt data using RSA

public static byte[] encrypt(String plaintext, PublicKey publicKey) throws Exception {

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

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

return cipher.doFinal(plaintext.getBytes());

}

// Method to decrypt data using RSA

public static String decrypt(byte[] ciphertext, PrivateKey privateKey) throws Exception {

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

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

byte[] decryptedBytes = cipher.doFinal(ciphertext);

return new String(decryptedBytes);

}

// Method to print the details of the RSA keys

public static void printKeyDetails(PublicKey publicKey, PrivateKey privateKey) throws Exception {

KeyFactory keyFactory = KeyFactory.getInstance("RSA");

RSAPublicKeySpec publicKeySpec = keyFactory.getKeySpec(publicKey, RSAPublicKeySpec.class);

RSAPrivateKeySpec privateKeySpec = keyFactory.getKeySpec(privateKey, RSAPrivateKeySpec.class);

System.out.println("Public Key Modulus: " + publicKeySpec.getModulus());

System.out.println("Public Key Exponent: " + publicKeySpec.getPublicExponent());

System.out.println("Private Key Modulus: " + privateKeySpec.getModulus());

System.out.println("Private Key Exponent: " + privateKeySpec.getPrivateExponent());

}

}

**OutPut:**

Public Key Modulus: 31038510711829999801784372531946983358137854990088090991637390148419287874509153732622733244983277737425077411478209840595153629807540368384755355846411296224073599759101428383921588840433014511099978908292086199561820753702801788517759664227014046917265694990286357338266729729289759181705824298589636392653898215407107629519168969960520004554964847765472617712501675590258013735930876058192220554294491498130406733122672893286235649989513594615421730088151244710489381704902227862207963934076878865541492434417834379610753508880734215780473233236081411679383688039381465088647599564440412214516283780812790414217103

Public Key Exponent: 65537

Private Key Modulus: 31038510711829999801784372531946983358137854990088090991637390148419287874509153732622733244983277737425077411478209840595153629807540368384755355846411296224073599759101428383921588840433014511099978908292086199561820753702801788517759664227014046917265694990286357338266729729289759181705824298589636392653898215407107629519168969960520004554964847765472617712501675590258013735930876058192220554294491498130406733122672893286235649989513594615421730088151244710489381704902227862207963934076878865541492434417834379610753508880734215780473233236081411679383688039381465088647599564440412214516283780812790414217103

Private Key Exponent: 8087242457164793576380822212727568973611273201561625368466668815698121057496045121660524480695400318053475470015440304530308732206136370760609769388792884848593630918205227445318599432980364615279052136014704120477251799597617274832983872107977049073946457842960920364194923125217082377692122857663863634907791753677319972893013752261852766476312175132103973308748602166897127203181256109741811130859412418599295121918297878348401549128762136412945086917802636900893271776649908071001640098967859925993336944084150579568040937047234823727319352079512156028482589697469338705761831391191660030457857029384955466071617

Original Text: Hello, RSA!

Encrypted Text: ??\_x0019\_­????~?6?91DQv<?\_x0012\_)

(C"???XF?K?J?%?A?o???\_x001D\_‑B61??1N

?O

?2?

?????K???0

Decrypted Text: Hello, RSA!

=== Code Execution Successful ===

9.**Write a java program to calculate the message digest of text using the SHA-1 algorithm?**

**Program:**

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class SHA1DigestExample {

public static void main(String[] args) {

String input = "Hello, World!"; // The input text for which SHA-1 hash is to be calculated

try {

// Create a MessageDigest instance for SHA-1

MessageDigest md = MessageDigest.getInstance("SHA-1");

// Update the MessageDigest with the bytes of the input string

md.update(input.getBytes());

// Perform the hash computation and get the resulting byte array

byte[] digest = md.digest();

// Convert the byte array into a hexadecimal string

StringBuilder sb = new StringBuilder();

for (byte b : digest) {

sb.append(String.format("%02x", b));

}

// Print the resulting SHA-1 hash

System.out.println("SHA-1 Digest: " + sb.toString());

} catch (NoSuchAlgorithmException e) {

System.out.println("SHA-1 algorithm not found: " + e.getMessage());

}

}

}

**OutPut:**

SHA-1 Digest: 0a0a9f2a6772942557ab5355d76af442f8f65e01

=== Code Execution Successful ===

**10.Write a java program to calculate the message digest of text using the MD5 algorithm?**

**Program:**

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class MD5DigestExample {

public static void main(String[] args) {

String input = "Hello, World!"; // The input text for which MD5 hash is to be calculated

try {

// Create a MessageDigest instance for MD5

MessageDigest md = MessageDigest.getInstance("MD5");

// Update the MessageDigest with the bytes of the input string

md.update(input.getBytes());

// Perform the hash computation and get the resulting byte array

byte[] digest = md.digest();

// Convert the byte array into a hexadecimal string

StringBuilder sb = new StringBuilder();

for (byte b : digest) {

sb.append(String.format("%02x", b));

}

// Print the resulting MD5 hash

System.out.println("MD5 Digest: " + sb.toString());

} catch (NoSuchAlgorithmException e) {

System.out.println("MD5 algorithm not found: " + e.getMessage());

}

}

}

**OutPut:**

MD5 Digest: 65a8e27d8879283831b664bd8b7f0ad4

=== Code Execution Successful ===