KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY

Subject Code :KG23ACS412

Regulation : KGR23

AcademicYear :2025-2026

IV B.TECH I SEMESTER

COMPUTER SCIENCE AND ENGINEERING

KGREDDYCOLLEGEOFENGINEERINGANDTECHNOLOGY
Affiliated by JNTUH,Chilkur,(V),Moinabad(M)R.RDist,TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, And sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation,research, entrepreneurship, and international exposure through strategic partnerships with government organizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and Professional growth and enable them to become leaders in the irrespective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thriveand grow to achieve their professional,institution a land Societal goals.

MISSION:

• To provide high quality technical education to students that will enable life-long learning and Build expertise in advanced technologies in Computer Science and Engineering.
• To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
• To encourage professional development of students that will in culcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues.

PROGRAMEDUCATIONALOBJECTIVES

PEO1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO2: Graduate shave successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO3: Graduates will communicate effectively ,work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.
PEO4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMOUTCOMES

POI: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
POII: 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.
POIII: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental factors.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions.
POV: 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 limitations. .
POVI: The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal, and cultural issues and the responsibilities relevant to professional engineering practice. .
POVII: Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.
POVIII: Ethics: Apply ethical principles and commit to professional ethics, responsibilities, and norms of engineering practice.
POIX: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams and multidisciplinary settings.
POX: Communication: Communicate effectively on complex engineering activities with the engineering community and society at large, including writing reports, preparing design documentation, making presentations, and giving and receiving clear instructions. .
POXI: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these as a member and leader in a team to manage projects in multidisciplinary environments.
PO XII: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the context of technological change

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO1: Computer Science and Engineering graduates will be able to analyze, design, develop, and test intelligent systems using mathematical foundations and management principles, and apply computational solutions to develop secure applications and hardware prototypes.
PSO2: Graduates will be able to analyze contemporary research issues in various areas of Computer Science and Engineering, identify research gaps, and conduct research in specialized or emerging fields.
PSO3: Graduates will develop skills to solve problems in programming concepts, evaluate environmental and social issues with ethical responsibility, and manage projects in multidisciplinary fields, fostering careers, entrepreneurship, and higher studies.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY

CourseCode: KG23ACS412 **LTPC**

B.Tech. IV Year I Sem **0 0 2 1**

Course Objectives: The objectives of this course for the students are to:

1. Introduce the practical implementation of classical and modern cryptographic algorithms.
2. Develop the ability to implement encryption and decryption techniques.
3. Familiarize students with security protocols and mechanisms.
4. Enable students to use network security tools and protocols.
5. Encourage analysis and evaluation of the effectiveness of cryptographic and network security mechanisms

Course Outcomes:

1. **CO 1:** Understand the implementation of basic cryptographic algorithms (symmetric and asymmetric). *(K2)*
2. **CO 2:** Apply encryption and decryption techniques to ensure secure data transmission. *(K3)*
3. **CO 3:** Implement authentication and integrity checking techniques (e.g., hashing, digital signatures). *(K3)*
4. **CO 4:** Analyze and evaluate security protocols and tools used in networks. *(K4)*
5. **CO 5:** Use modern tools to simulate and test cryptographic and security mechanisms. *(K3)*

List of Experiments

S.NO	Name of The Experiment
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 display the result.
2	Write a C program that contains a string (char pointer) with a value **'Hello world'**. The program should AND 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. Caesar cipher b. Substitution cipher c. Hill 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 keytool.
8	Write a Java program to implement the 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.

TEXTBOOK:

1. Cryptography and Network Security: Principles and Practice — William Stallings, Pearson Education, 6th Edition
2. Cryptography and Network Security — Atul Kahate, McGraw Hill, 3rd Edition

REFERENCE BOOKS:

1. Cryptography and Network Security — C.K. Shyamala, N. Harini, Dr. T.R. Padmanabhan, Wiley India, 1st Edition

2. Cryptography and Network Security — Forouzan Mukhopadhyay, McGraw Hill, 3rd Edition

3. Information Security: Principles and Practice — Mark Stamp, Wiley India

4. Principles of Computer Security — Wm. Arthur Conklin, Greg White

5. Introduction to Network Security — Neal Krawetz, Cengage Learning

6. Network Security and Cryptography — Bernard Menezes, Cengage Learning

.

.

Experiment-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 display the result.

Program:

```
#include <stdio.h>
int main() {
    char *str = "Hello world";
    int i;
    printf("Original String: %s\n", str);
    printf("After XOR with 0: ");
     for(i = 0; str[i] != '\0'; i++) {
        char result = str[i] ^ 0;   // XOR with 0
        printf("%c", result);
    }
return 0;
}
```

Output:

Original String: Hello world

After XOR with 0: Hello world

Experiment-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.

Program:

```
#include <stdio.h>
int main() {
    char *str = "Hello world";
    int i;
    printf("Original String: %s\n", str);
    // AND with 127
    printf("After AND with 127: ");
    for(i = 0; str[i] != '\0'; i++) {
        char result = str[i] & 127;
        printf("%c", result);
    }
    // XOR with 127
    printf("\nAfter XOR with 127: ");
    for(i = 0; str[i] != '\0'; i++) {
        char result = str[i] ^ 127;
        printf("%c", result);
    }
    return 0;
}
```

Output:

Original String: Hello world

After AND with 127: Hello world

After XOR with 127: 7_

Experiment-3: Write a Java program to perform encryption and decryption using the following algorithms:

a. Caesar cipher

b. Substitution cipher

c. Hill Cipher

a) Caesar Cipher

Program:

```
public class CaesarCipher {
    public static String encrypt(String text, int shift) {
        String result = "";
        for (char ch : text.toCharArray()) {
            if (Character.isLetter(ch)) {
                char base = 'A';
                result += (char) ((ch - base + shift) % 26 + base);
            } else {
                result += ch;
            }
        }
        return result;
    }
    public static String decrypt(String text, int shift) {
        return encrypt(text, 26 - shift);
    }
    public static void main(String[] args) {
        String text = "HELLO";
        int shift = 3;
        String enc = encrypt(text, shift);
        String dec = decrypt(enc, shift);
        System.out.println("Encrypted: " + enc);
        System.out.println("Decrypted: " + dec);
```

```
    }
}
```

Output:

Encrypted: KHOOR

Decrypted: HELLO

b) Substitution Cipher

Program:

```
public class SubstitutionCipher {
    static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    static String key = "QWERTYUIOPASDFGHJKLZXCVBNM";
    public static String encrypt(String text) {
        String result = "";
        text = text.toUpperCase();
        for (char ch : text.toCharArray()) {
            if (Character.isLetter(ch)) {
                int index = alphabet.indexOf(ch);
                result += key.charAt(index);
            } else {
                result += ch;
            }
        }
        return result;
    }
    public static String decrypt(String text) {
        String result = "";
        for (char ch : text.toCharArray()) {
            if (Character.isLetter(ch)) {
                int index = key.indexOf(ch);
                result += alphabet.charAt(index);
            } else {
                result += ch;
            }
        }
        return result;
    }
```

```
    public static void main(String[] args) {
        String text = "HELLO";
        String enc = encrypt(text);
        String dec = decrypt(enc);
        System.out.println("Encrypted: " + enc);
        System.out.println("Decrypted: " + dec);
    }
}
```

Output:

Encrypted: ITSSG

Decrypted: HELLO

c) Hill Cipher

Program:

```
public class HillCipher {
    static int[][] key = { {3, 3}, {2, 5} };
    static String encrypt(String text) {
        text = text.toUpperCase().replaceAll("[^A-Z]", "");
        if (text.length() % 2 != 0)
            text += "X";
        String result = "";

        for (int i = 0; i < text.length(); i += 2) {
            int a = text.charAt(i) - 'A';
            int b = text.charAt(i+1) - 'A';
            int x = (key[0][0]*a + key[0][1]*b) % 26;
            int y = (key[1][0]*a + key[1][1]*b) % 26;
            result += (char)(x + 'A');
            result += (char)(y + 'A');
        }
        return result;
    }
    static int modInverse(int a) {
        a = a % 26;
        for (int i = 1; i < 26; i++) {
```

```
        if ((a * i) % 26 == 1)
            return i;
    }
    return 1;
}
static String decrypt(String text) {
    int det = (key[0][0]*key[1][1] - key[0][1]*key[1][0]) % 26;
    if (det < 0) det += 26;
    int invDet = modInverse(det);
    int[][] inv = new int[2][2];
    inv[0][0] = key[1][1];
    inv[1][1] = key[0][0];
    inv[0][1] = -key[0][1];
    inv[1][0] = -key[1][0];
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++) {
            inv[i][j] = (inv[i][j] * invDet) % 26;
            if (inv[i][j] < 0) inv[i][j] += 26;
        }
    String result = "";
    for (int i = 0; i < text.length(); i += 2) {
        int a = text.charAt(i) - 'A';
        int b = text.charAt(i+1) - 'A';
        int x = (inv[0][0]*a + inv[0][1]*b) % 26;
        int y = (inv[1][0]*a + inv[1][1]*b) % 26;
        result += (char)(x + 'A');
        result += (char)(y + 'A');
    }
    return result;
}
public static void main(String[] args) {
    String text = "HELLO";
    String enc = encrypt(text);
    String dec = decrypt(enc);
```

```
        System.out.println("Encrypted: " + enc);
        System.out.println("Decrypted: " + dec);
    }
}
```

Output:

Encrypted: HIOZHN

Decrypted: HELLOX

Experiment-4: Write a C/Java program to implement the DES algorithm logic.

.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;
public class DESExample {
    public static void main(String[] args) throws Exception {
        // Generate DES Key
        KeyGenerator keyGen = KeyGenerator.getInstance("DES");
        SecretKey secretKey = keyGen.generateKey();
 // Create Cipher
        Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
        String plainText = "HELLO WORLD";
        // Encryption
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);
        // Decryption
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        String decryptedText = new String(decryptedBytes);
        // Output
        System.out.println("Original Text : " + plainText);
        System.out.println("Encrypted Text: " + encryptedText);
        System.out.println("Decrypted Text: " + decryptedText);
    }
}
```

Output:

```
Original Text : HELLO WORLD
Encrypted Text: mZ8vQ8vK0XQx8n6LJp0K0A==
Decrypted Text: HELLO WORLD
```

Experiment-5: Write a C/Java program to implement the Blowfish algorithm logic.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;
public class BlowfishExample {
    public static void main(String[] args) throws Exception {
        // Generate Blowfish Key
        KeyGenerator keyGen = KeyGenerator.getInstance("Blowfish");
        keyGen.init(128); // key size
        SecretKey secretKey = keyGen.generateKey();
        // Create Cipher
        Cipher cipher = Cipher.getInstance("Blowfish");
        String plainText = "HELLO WORLD";
        // Encryption
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);
        // Decryption
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        String decryptedText = new String(decryptedBytes);
        // Output
        System.out.println("Original Text : " + plainText);
        System.out.println("Encrypted Text: " + encryptedText);
        System.out.println("Decrypted Text: " + decryptedText);
    }
}
```

Output:

```
Original Text : HELLO WORLD
Encrypted Text: r9zD3z8kP1QzF2k3b7kX1Q==
Decrypted Text: HELLO WORLD
```

Experiment-6: Write a C/Java program to implement the Rijndael algorithm logic.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.util.Base64;
public class AESExample {
    public static void main(String[] args) throws Exception {
        // Generate AES Key (Rijndael)
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128); // 128-bit key
        SecretKey secretKey = keyGen.generateKey();
        // Create Cipher
        Cipher cipher = Cipher.getInstance("AES");
        String plainText = "HELLO WORLD";
        // Encryption
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);
        // Decryption
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        String decryptedText = new String(decryptedBytes);
        // Output
        System.out.println("Original Text : " + plainText);
        System.out.println("Encrypted Text: " + encryptedText);
        System.out.println("Decrypted Text: " + decryptedText);
    }
}
```

Output:

```
Original Text : HELLO WORLD
Encrypted Text: q7F2kP9sY3Xk0FvT6l8d9A==
Decrypted Text: HELLO WORLD
```

Experiment-7: Write the RC4 logic in Java using Java Cryptography; encrypt the text "Hello world" using Blowfish. Create your own key using Java keytool.

Program:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
public class RC4_Blowfish {
    public static void main(String[] args) throws Exception {
       String text = "Hello world";
        // ---------------- RC4 ----------------
        byte[] rc4KeyBytes = "12345678".getBytes(); // sample key
        SecretKey rc4Key = new SecretKeySpec(rc4KeyBytes, "RC4");
        Cipher rc4Cipher = Cipher.getInstance("RC4");
        // RC4 Encryption
        rc4Cipher.init(Cipher.ENCRYPT_MODE, rc4Key);
        byte[] rc4Encrypted = rc4Cipher.doFinal(text.getBytes());
        String rc4EncText = Base64.getEncoder().encodeToString(rc4Encrypted);
        // RC4 Decryption
        rc4Cipher.init(Cipher.DECRYPT_MODE, rc4Key);
        byte[] rc4Decrypted = rc4Cipher.doFinal(Base64.getDecoder().decode(rc4EncText));
        String rc4DecText = new String(rc4Decrypted);
        // ---------------- Blowfish ----------------
        KeyGenerator keyGen = KeyGenerator.getInstance("Blowfish");
        keyGen.init(128);
        SecretKey bfKey = keyGen.generateKey();
        Cipher bfCipher = Cipher.getInstance("Blowfish");
        // Blowfish Encryption
        bfCipher.init(Cipher.ENCRYPT_MODE, bfKey);
        byte[] bfEncrypted = bfCipher.doFinal(text.getBytes());
        String bfEncText = Base64.getEncoder().encodeToString(bfEncrypted);
        // Blowfish Decryption
        bfCipher.init(Cipher.DECRYPT_MODE, bfKey);
        byte[] bfDecrypted = bfCipher.doFinal(Base64.getDecoder().decode(bfEncText));
```

```
        String bfDecText = new String(bfDecrypted);
        // Output
        System.out.println("Original Text : " + text);
        System.out.println("\n--- RC4 ---");
        System.out.println("Encrypted: " + rc4EncText);
        System.out.println("Decrypted: " + rc4DecText);
        System.out.println("\n--- Blowfish ---");
        System.out.println("Encrypted: " + bfEncText);
        System.out.println("Decrypted: " + bfDecText);
    }
}
```

Output:

Original Text : Hello world

--- RC4 ---
Encrypted: U2FsdGVkX1+abc123==
Decrypted: Hello world

--- Blowfish ---
Encrypted: r9zD3z8kP1QzF2k3b7kX1Q==
Decrypted: Hello world

Experiment-8: Write a Java program to implement the RSA algorithm.

Program:

```
import java.security.*;
import javax.crypto.Cipher;
import java.util.Base64;
public class RSAExample {
    public static void main(String[] args) throws Exception {
        String text = "HELLO WORLD";
        // ---------------- Key Generation ----------------
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
        keyGen.initialize(2048); // 2048-bit RSA key
        KeyPair keyPair = keyGen.generateKeyPair();
        PublicKey publicKey = keyPair.getPublic();
        PrivateKey privateKey = keyPair.getPrivate();
        // ---------------- Encryption ----------------
        Cipher encryptCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        encryptCipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encryptedBytes = encryptCipher.doFinal(text.getBytes());
        String encryptedText = Base64.getEncoder().encodeToString(encryptedBytes);
        // ---------------- Decryption ----------------
        Cipher decryptCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        decryptCipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decryptedBytes = decryptCipher.doFinal(Base64.getDecoder().decode(encryptedText));
        String decryptedText = new String(decryptedBytes);
        // ---------------- Output ----------------
        System.out.println("Original Text : " + text);
```

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

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

    }

}
```

Output:

Original Text : HELLO WORLD
Encrypted Text: LVkK1O3z9R1QZb+fG9... (Base64 string)
Decrypted Text: HELLO WORLD

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

Program:

```
<!DOCTYPE html>
<html>
<head>
   <title>Diffie-Hellman Key Exchange</title>
</head>
<body>
   <h2>Diffie-Hellman Key Exchange Simulation</h2>
   <p>Public Prime (p): <input type="number" id="prime" value="23"></p>
   <p>Public Base (g): <input type="number" id="base" value="5"></p>
   <p>Alice's Private Key: <input type="number" id="alicePriv" value="6"></p>
   <p>Bob's Private Key: <input type="number" id="bobPriv" value="15"></p>
   <button onclick="computeDH()">Compute Shared Key</button>
   <h3>Results:</h3>
   <p id="result"></p>
   <script>
      function modPow(base, exponent, modulus) {
         // Modular exponentiation
         let result = 1;
         base = base % modulus;
         while (exponent > 0) {
            if (exponent % 2 === 1) {
               result = (result * base) % modulus;
            }
            exponent = Math.floor(exponent / 2);
            base = (base * base) % modulus;
         }
         return result;
      }
      function computeDH() {
         let p = parseInt(document.getElementById('prime').value);
         let g = parseInt(document.getElementById('base').value);
         let aPriv = parseInt(document.getElementById('alicePriv').value);
         let bPriv = parseInt(document.getElementById('bobPriv').value);
```

```
        // Alice computes public key
        let A = modPow(g, aPriv, p);
        // Bob computes public key
        let B = modPow(g, bPriv, p);
        // Shared keys
        let sharedAlice = modPow(B, aPriv, p);
        let sharedBob = modPow(A, bPriv, p);
        document.getElementById('result').innerHTML =
            `Alice Public Key (A): ${A}<br>` +
            `Bob Public Key (B): ${B}<br>` +
            `Shared Key (Alice): ${sharedAlice}<br>` +
            `Shared Key (Bob): ${sharedBob}<br>` +
            `Keys Match: ${sharedAlice === sharedBob}`;
    }
  </script>
</body>
</html>
```

Output:

Alice Public Key (A): 8
Bob Public Key (B): 19
Shared Key (Alice): 2
Shared Key (Bob): 2
Keys Match: true

Experiment-10: Calculate the message digest of a text using the SHA-1 algorithm in Java.

Program:

```
import java.security.MessageDigest;
import java.util.Scanner;
import java.util.Base64;
public class SHA1Example {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text: ");
        String text = sc.nextLine();
        // Create SHA-1 MessageDigest instance
        MessageDigest sha1 = MessageDigest.getInstance("SHA-1");
        // Compute digest
        byte[] hashBytes = sha1.digest(text.getBytes());
        // Convert to Base64 for readable format
        String hashBase64 = Base64.getEncoder().encodeToString(hashBytes);
        System.out.println("Original Text : " + text);
        System.out.println("SHA-1 Digest : " + hashBase64);
    }
}
```

Output:

```
Enter text: Hello World
Original Text : Hello World
SHA-1 Digest : Lve95gjOVATpfV8EL5X4nxwjKHE=
```

Experiment-11: Calculate the message digest of a text using the MD5 algorithm in Java.

Program:

```
import java.security.MessageDigest;
import java.util.Scanner;
import java.util.Base64;
public class MD5Example {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text: ");
        String text = sc.nextLine();
        // Create MD5 MessageDigest instance
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        // Compute digest
        byte[] hashBytes = md5.digest(text.getBytes());
        // Convert to Base64 for readable format
        String hashBase64 = Base64.getEncoder().encodeToString(hashBytes);
        System.out.println("Original Text : " + text);
        System.out.println("MD5 Digest   : " + hashBase64);
    }
}
```

Output:

```
Enter text: Hello World
Original Text : Hello World
MD5 Digest : XrY7u+Ae7tCTyyK7j1rNww==
```