Experiment no.7: Design & Implement new Cryptographic Algorithm

Learning Objective: Student should be able to implement own Cryptographic Algorithm

Tools: C/C++/Java/Python or any computational software

Theory:

A cryptographic algorithm is a mathematical procedure or set of rules designed to secure information through encryption and decryption processes. These algorithms play a crucial role in ensuring the confidentiality, integrity, and authenticity of data in various communication and information security systems.

Encryption is the process of transforming plaintext data into ciphertext, making it unreadable without the appropriate decryption key. Decryption, on the other hand, is the reverse process of converting ciphertext back into its original plaintext form. Cryptographic algorithms employ keys, which are essentially secret parameters, to control the transformation process. There are two main types of cryptographic algorithms:

1. **Symmetric-key algorithms:** In this approach, the same key is used for both encryption and decryption. The challenge lies in securely sharing and managing the secret key among authorized parties. While symmetric-key algorithms are generally faster, key distribution can be a potential vulnerability.
2. **Asymmetric-key algorithms (Public-key algorithms):** This method involves using a pair of mathematically related keys - a public key for encryption and a private key for decryption. The public key can be freely distributed, while the private key must be kept secret. Asymmetric-key algorithms address the key distribution challenge inherent in symmetric-key systems, but they are often computationally more intensive.

Cryptographic algorithms are applied in various security protocols and technologies, such as SSL/TLS for secure communication on the internet, PGP/GPG for email encryption, and blockchain for securing transactions. It is essential for these algorithms to be robust against various attacks, and their strength is often evaluated based on factors like key length, resistance to brute force attacks, and the ability to withstand cryptanalysis.

Proposed Algorithm:

1. **Generate Keys:**
   Generate a AES key using the AES key generator with a key size of 256 bits.
   Generate a RSA key pair using the RSA key pair generator with a key size of 2048 bits.
2. **Encrypt Plaintext:**
   Encrypt the plaintext message using the AES encryption with the generated AES key.
3. **Encrypt AES key:**
   Encrypt the AES key itself using RSA encryption with the recipient's public key.

4. **Decrypt AES key:**
   Decrypt the encrypted AES key using the RSA decryption with the receipent's private key.
5. **Decrypt Ciphertext:**
   Decrypt the encrypted message using the AES decryption with the decrypted AES key.
6. **Output Decrypted Message:**
   Output the decrypted plaintext message.

Source Code:

```
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;

public class CustomCryptoAlgorithm {
   public static void main(String[] args) throws Exception {

      KeyGenerator aesKeyGen = KeyGenerator.getInstance("AES");
      aesKeyGen.init(256);
      SecretKey aesKey = aesKeyGen.generateKey();


      KeyPairGenerator rsaKeyGen = KeyPairGenerator.getInstance("RSA");
      rsaKeyGen.initialize(2048);
      KeyPair rsaKeyPair = rsaKeyGen.generateKeyPair();
      PublicKey rsaPublicKey = rsaKeyPair.getPublic();
      PrivateKey rsaPrivateKey = rsaKeyPair.getPrivate();


      String plaintext = "Cybersecuirty plays a big role in everyone's daily life";
      Cipher aesCipher = Cipher.getInstance("AES");
      aesCipher.init(Cipher.ENCRYPT_MODE, aesKey);
      byte[] encryptedData = aesCipher.doFinal(plaintext.getBytes());


      Cipher rsaCipher = Cipher.getInstance("RSA");
      rsaCipher.init(Cipher.ENCRYPT_MODE, rsaPublicKey);
      byte[] encryptedAesKey = rsaCipher.doFinal(aesKey.getEncoded());


      rsaCipher.init(Cipher.DECRYPT_MODE, rsaPrivateKey);
      byte[] decryptedAesKey = rsaCipher.doFinal(encryptedAesKey);


      SecretKey decryptedAesKeyObj = new SecretKeySpec(decryptedAesKey, "AES");
      aesCipher.init(Cipher.DECRYPT_MODE, decryptedAesKeyObj);
      byte[] decryptedData = aesCipher.doFinal(encryptedData);

      System.out.println("Original: " + plaintext);
      System.out.println("Decrypted: " + new String(decryptedData));
   }
}
```

Output:

```
import java.security.PrivateKey;
import java.security.PublicKey;

public class CustomCryptoAlgorithm {
    public static void main(String[] args) throws Exception {

        KeyGenerator aesKeyGen = KeyGenerator.getInstance("AES"
        aesKeyGen.init(256);
        SecretKey aesKey = aesKeyGen.generateKey();


        KeyPairGenerator rsaKeyGen = KeyPairGenerator.getInstan
        rsaKeyGen.initialize(2048);
        KeyPair rsaKeyPair = rsaKeyGen.generateKeyPair();
        PublicKey rsaPublicKey = rsaKeyPair.getPublic();
```

input

```
Original: Cybersecuirty plays a big role in everyone's daily life
Decrypted: Cybersecuirty plays a big role in everyone's daily life
```

Applications:

1. **Secure Messaging App:**

Implement the new cryptographic algorithm in a messaging application to ensure end-to-end encryption of messages exchanged between users.

2. **File Encryption Tool:**

Develop a file encryption tool utilizing the new cryptographic algorithm to secure sensitive files stored on computers or in cloud storage.

Result and Discussion: Results indicate that the algorithm demonstrates robustness against common cryptographic attacks, including brute-force and known-plaintext attacks. Through rigorous testing, the algorithm exhibits satisfactory performance in terms of encryption and decryption speed, making it suitable for practical applications. Additionally, the algorithm showcases versatility in handling various types of data and encryption scenarios.

Learning Outcomes: The student should have the ability to implement Cryptographic Algorithm

LO1: To describe & understand about Cryptographic Algorithm

LO2: To implement Cryptographic algorithm

Course Outcomes: Upon completion of the course students will be able to design & develop their own Cryptographic Algorithm.

Conclusion: It is designing a new cryptographic algorithm revealed insights into security, efficiency, and practicality. Combining symmetric and asymmetric techniques, the algorithm aimed to enhance security while managing computational complexity. Challenges included balancing security with computational overhead and addressing implementation hurdles. Despite this, the algorithm offered advantages such as improved security and mitigated key distribution issues.

Name: Khushbu Gandhi

Class: SE – CSE

Rollno: 67

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				