



INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

ALUMNO:

ANGEL DE JESÚS FISCAL MALAGA

MATERIA:

CRIPTOGRAFÍA

CARRERA:

INGENIERIA INFORMATICA

TEMA:

EXAMEN UNIDAD 3

PROFESOR:

ERICK TELLEZ VERA

GRUPO: 910 B

FECHA:

SAN ANDRÉS TUXTLA, VER. A 23 DE NOVIEMBRE DEL 2025

MAIN APP.java

```
package com.cifrador.ui;

import com.cifrador.database.DatabaseHelper;
import com.cifrador.encryptor.HybridCipher;
import com.cifrador.utils.RSAUtil;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
import java.security.KeyPair;
import java.security.PrivateKey;
import java.security.PublicKey;

public class MainApp extends Application {

    private File selectedFileEncrypt;
    private File selectedFileDecrypt;
    private PublicKey loadedPublicKey;
    private PrivateKey loadedPrivateKey;

    private Label status;
    private Label publicKeyLabel, privateKeyLabel, fileLabelEncrypt,
fileLabelDecrypt;
    private TextField userField;

    private final String DB_HOST = "localhost";
    private final int DB_PORT = 3306;
    private final String DB_NAME = "cifrador";
    private final String DB_USER = "malaga";
    private final String DB_PASSWORD = "malaga";

    private DatabaseHelper dbHelper;

    @Override
    public void start(Stage primaryStage) {
```

```

dbHelper = new DatabaseHelper(DB_HOST, DB_PORT, DB_NAME,
DB_USER, DB_PASSWORD);
primaryStage.setTitle("Cifrador Híbrido RSA-AES");

status = new Label("Estado: listo.");

Label titulo1 = new Label("1.- Usuario Nuevo");
titulo1.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

Label txtNuevoUser = new Label("¿Eres usuario nuevo? Regístrate aquí");
userField = new TextField();
userField.setPromptText("Ingresa nombre de usuario");

Button genKeysBtn = new Button("Generar par RSA y guardar");
genKeysBtn.setOnAction(e -> handleGenerateKeys(primaryStage));

VBox col1 = new VBox(10, titulo1, txtNuevoUser, userField, genKeysBtn);
col1.setAlignment(Pos.TOP_CENTER);
col1.setPadding(new Insets(20));

Label titulo2 = new Label("2.- Cifrar Archivo");
titulo2.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

Label lblPub = new Label("Carga tu clave pública");
Button loadPubBtn = new Button("Seleccionar clave pública");
publicKeyLabel = new Label("Ninguna clave cargada");
loadPubBtn.setOnAction(e -> handleLoadPublicKey(primaryStage));

Label lblFileEnc = new Label("Selecciona tu archivo");
Button chooseFileEncBtn = new Button("Seleccionar archivo");
fileLabelEncrypt = new Label("Ningún archivo seleccionado");
chooseFileEncBtn.setOnAction(e -> selectedFileEncrypt =
handleSelectFile(primaryStage, fileLabelEncrypt));

Label lblCifrar = new Label("Dale en Cifrar archivo");
Button encryptBtn = new Button("Cifrar archivo");
encryptBtn.setOnAction(e -> handleEncrypt());

VBox col2 = new VBox(10, titulo2, lblPub, loadPubBtn, publicKeyLabel,
lblFileEnc, chooseFileEncBtn, fileLabelEncrypt, lblCifrar, encryptBtn);
col2.setAlignment(Pos.TOP_CENTER);
col2.setPadding(new Insets(20));

```

```

Label titulo3 = new Label("3.- Descifrar Archivo");
titulo3.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");

Label lblPriv = new Label("Carga tu clave privada");
Button loadPrivBtn = new Button("Seleccionar clave privada");
privateKeyLabel = new Label("Ninguna clave cargada");
loadPrivBtn.setOnAction(e -> handleLoadPrivateKey(primaryStage));

Label lblFileDec = new Label("Selecciona tu archivo cifrado");
Button chooseFileDecBtn = new Button("Seleccionar archivo");
fileLabelDecrypt = new Label("Ningún archivo seleccionado");
chooseFileDecBtn.setOnAction(e -> selectedFileDecrypt =
handleSelectFile(primaryStage, fileLabelDecrypt));

Label lblDescifrar = new Label("Dale en Descifrar archivo");
Button decryptBtn = new Button("Descifrar archivo");
decryptBtn.setOnAction(e -> handleDecrypt());

VBox col3 = new VBox(10, titulo3, lblPriv, loadPrivBtn, privateKeyLabel,
    lblFileDec, chooseFileDecBtn, fileLabelDecrypt, lblDescifrar, decryptBtn);
col3.setAlignment(Pos.TOP_CENTER);
col3.setPadding(new Insets(20));

GridPane root = new GridPane();
root.setPadding(new Insets(20));
root.setHgap(30);
root.setAlignment(Pos.CENTER);

root.add(col1, 0, 0);
root.add(col2, 1, 0);
root.add(col3, 2, 0);

VBox mainLayout = new VBox(10, root, status);
mainLayout.setAlignment(Pos.CENTER);
mainLayout.setPadding(new Insets(10));

Scene scene = new Scene(mainLayout, 950, 500);
primaryStage.setScene(scene);
primaryStage.show();
}

private void handleGenerateKeys(Stage stage) {
    try {

```

```

    KeyPair kp = RSAUtil.generateKeyPair(2048);

    FileChooser fc = new FileChooser();
    fc.setTitle("Guardar clave pública");
    fc.setInitialFileName("public_key.pem");
    File pubFile = fc.showSaveDialog(stage);
    if (pubFile != null) RSAUtil.savePublicKey(kp.getPublic(), pubFile);

    FileChooser fc2 = new FileChooser();
    fc2.setTitle("Guardar clave privada");
    fc2.setInitialFileName("private_key.pem");
    File privFile = fc2.showSaveDialog(stage);
    if (privFile != null) RSAUtil.savePrivateKey(kp.getPrivate(), privFile);

    status.setText("Par de claves RSA generado y guardado.");
} catch (Exception ex) {
    status.setText("Error generando claves: " + ex.getMessage());
}
}

private void handleLoadPublicKey(Stage stage) {
    FileChooser fc = new FileChooser();
    fc.setTitle("Seleccionar clave pública (PEM)");
    File f = fc.showOpenDialog(stage);
    if (f == null) return;
    try {
        loadedPublicKey = RSAUtil.loadPublicKey(f);
        publicKeyLabel.setText("Clave pública cargada: " + f.getName());
    } catch (Exception ex) {
        status.setText("Error al cargar clave pública: " + ex.getMessage());
    }
}

private File handleSelectFile(Stage stage, Label fileLabel) {
    FileChooser fc = new FileChooser();
    fc.setTitle("Seleccionar archivo");
    File f = fc.showOpenDialog(stage);
    if (f != null) fileLabel.setText("Archivo seleccionado: " + f.getName());
    return f;
}

private void handleLoadPrivateKey(Stage stage) {
    FileChooser fc = new FileChooser();

```

```

        fc.setTitle("Seleccionar clave privada (PEM)");
        File f = fc.showOpenDialog(stage);
        if (f == null) return;
        try {
            loadedPrivateKey = RSAUtil.loadPrivateKey(f);
            privateKeyLabel.setText("Clave privada cargada: " + f.getName());
        } catch (Exception ex) {
            status.setText("Error al cargar clave privada: " + ex.getMessage());
        }
    }

    private void handleEncrypt() {
        if (selectedFileEncrypt == null || loadedPublicKey == null) {
            status.setText("Selecciona archivo y clave pública antes de cifrar.");
            return;
        }
        try {
            File outEnc = new File(selectedFileEncrypt.getParentFile(),
selectedFileEncrypt.getName() + ".enc");
            File outKey = new File(selectedFileEncrypt.getParentFile(),
selectedFileEncrypt.getName() + ".key.enc");

            String claveCifradaB64 = HybridCipher.encryptFile(selectedFileEncrypt,
outEnc, outKey, loadedPublicKey);
            String usuario = userField.getText().trim();
            dbHelper.insertOperacion(selectedFileEncrypt.getName(),
claveCifradaB64, usuario.isEmpty() ? null : usuario);

            status.setText("Archivo cifrado correctamente: " + outEnc.getName());
        } catch (Exception ex) {
            status.setText("Error cifrando: " + ex.getMessage());
        }
    }

    private void handleDecrypt() {
        if (selectedFileDecrypt == null || loadedPrivateKey == null) {
            status.setText("Selecciona archivo cifrado y clave privada.");
            return;
        }
        try {
            File keyFile = new File(selectedFileDecrypt.getParentFile(),
selectedFileDecrypt.getName().replaceAll("\\.enc$", "") + ".key.enc");
            File out = new File(selectedFileDecrypt.getParentFile(),

```

```
        selectedFileDecrypt.getName().replaceAll("\\.enc$", "") + ".dec");

        HybridCipher.decryptFile(selectedFileDecrypt, keyFile, loadedPrivateKey,
out);
        status.setText("Archivo descifrado correctamente: " + out.getName());
    } catch (Exception ex) {
        status.setText("Error descifrando: " + ex.getMessage());
    }
}

public static void main(String[] args) {
    launch(args);
}
}
```

RSAUtil.java

```
package com.cifrador.utils;

import javax.crypto.Cipher;
import java.io.*;
import java.nio.file.Files;
import java.security.*;
import java.security.spec.*;
import java.util.Base64;

public class RSAUtil {
    public static final String RSA_ALGO = "RSA";
    public static final String RSA_TRANSFORMATION = "RSA/ECB/OAEPWithSHA-
256AndMGF1Padding";

    public static KeyPair generateKeyPair(int bits) throws Exception {
        KeyPairGenerator kpg = KeyPairGenerator.getInstance(RSA_ALGO);
        kpg.initialize(bits, SecureRandom.getInstanceStrong());
        return kpg.generateKeyPair();
    }

    public static void savePublicKey(PublicKey pub, File outFile) throws IOException
    {
        String b64 = Base64.getEncoder().encodeToString(pub.getEncoded());
        try (FileWriter fw = new FileWriter(outFile)) {
            fw.write("-----BEGIN PUBLIC KEY-----\n");
            fw.write(b64);
            fw.write("\n-----END PUBLIC KEY-----\n");
        }
    }

    public static void savePrivateKey(PublicKey priv, File outFile) throws
IOException {
        String b64 = Base64.getEncoder().encodeToString(priv.getEncoded());
        try (FileWriter fw = new FileWriter(outFile)) {
            fw.write("-----BEGIN PRIVATE KEY-----\n");
            fw.write(b64);
            fw.write("\n-----END PRIVATE KEY-----\n");
        }
    }

    public static PublicKey loadPublicKey(File f) throws Exception {
```



```

        String content = new String(Files.readAllBytes(f.toPath())).replaceAll("-----
.*KEY-----", "").replaceAll("\\s", "");
        byte[] decoded = Base64.getDecoder().decode(content);
        X509EncodedKeySpec spec = new X509EncodedKeySpec(decoded);
        KeyFactory kf = KeyFactory.getInstance(RSA_ALGO);
        return kf.generatePublic(spec);
    }

```

```

    public static PrivateKey loadPrivateKey(File f) throws Exception {
        String content = new String(Files.readAllBytes(f.toPath())).replaceAll("-----
.*KEY-----", "").replaceAll("\\s", "");
        byte[] decoded = Base64.getDecoder().decode(content);
        PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(decoded);
        KeyFactory kf = KeyFactory.getInstance(RSA_ALGO);
        return kf.generatePrivate(spec);
    }

```

```

    public static byte[] encryptWithPublicKey(byte[] data, PublicKey publicKey) throws
Exception {
        Cipher cipher = Cipher.getInstance(RSA_TRANSFORMATION);
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        return cipher.doFinal(data);
    }

```

```

    public static byte[] decryptWithPrivateKey(byte[] data, PrivateKey privateKey)
throws Exception {
        Cipher cipher = Cipher.getInstance(RSA_TRANSFORMATION);
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return cipher.doFinal(data);
    }
}

```

AESUtil.java

```
package com.cifrador.utils;

import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;
import java.io.*;
import java.security.SecureRandom;
import java.util.Base64;

public class AESUtil {
    public static final String AES_ALGO = "AES";
    public static final String AES_TRANSFORMATION = "AES/GCM/NoPadding";
    private static final int AES_KEY_SIZE = 256;
    private static final int GCM_IV_LENGTH = 12;
    private static final int GCM_TAG_LENGTH = 128;

    public static SecretKey generateKey() throws Exception {
        KeyGenerator kg = KeyGenerator.getInstance(AES_ALGO);
        kg.init(AES_KEY_SIZE, SecureRandom.getInstanceStrong());
        return kg.generateKey();
    }

    public static byte[] generateIV() {
        byte[] iv = new byte[GCM_IV_LENGTH];
        new SecureRandom().nextBytes(iv);
        return iv;
    }

    public static void encryptFile(File inFile, File outFile, SecretKey key, byte[] iv)
    throws Exception {
        Cipher cipher = Cipher.getInstance(AES_TRANSFORMATION);
        GCMParameterSpec spec = new GCMParameterSpec(GCM_TAG_LENGTH,
        iv);
        cipher.init(Cipher.ENCRYPT_MODE, key, spec);

        try (FileOutputStream fos = new FileOutputStream(outFile);
            CipherOutputStream cos = new CipherOutputStream(fos, cipher);
            FileInputStream fis = new FileInputStream(inFile)) {
            fos.write(iv);
        }
```

```

        byte[] buffer = new byte[4096];
        int r;
        while ((r = fis.read(buffer)) != -1) {
            cos.write(buffer, 0, r);
        }
    }
}

public static void decryptFile(File inFile, File outFile, SecretKey key) throws
Exception {
    try (FileInputStream fis = new FileInputStream(inFile)) {
        byte[] iv = new byte[GCM_IV_LENGTH];
        int read = fis.read(iv);
        if (read != GCM_IV_LENGTH) throw new IOException("IV no disponible o
archivo corrupto");

        Cipher cipher = Cipher.getInstance(AES_TRANSFORMATION);
        GCMParameterSpec spec = new
GCMParameterSpec(GCM_TAG_LENGTH, iv);
        cipher.init(Cipher.DECRYPT_MODE, key, spec);

        try (CipherInputStream cis = new CipherInputStream(fis, cipher);
            FileOutputStream fos = new FileOutputStream(outFile)) {
            byte[] buffer = new byte[4096];
            int r;
            while ((r = cis.read(buffer)) != -1) {
                fos.write(buffer, 0, r);
            }
        }
    }
}

public static String keyToBase64(SecretKey key) {
    return Base64.getEncoder().encodeToString(key.getEncoded());
}

public static SecretKey keyFromBase64(String b64) {
    byte[] decoded = Base64.getDecoder().decode(b64);
    return new javax.crypto.spec.SecretKeySpec(decoded, AES_ALGO);
}
}

```

HibridCypher.java

```
package com.cifrador.encryptor;

import com.cifrador.utils.AESUtil;
import com.cifrador.utils.RSAUtil;

import javax.crypto.SecretKey;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.util.Base64;

public class HybridCipher {

    public static String encryptFile(File inFile, File outEncFile, File outEncKeyFile,
        java.security.PublicKey publicKey) throws Exception {
        SecretKey aesKey = AESUtil.generateKey();
        byte[] iv = AESUtil.generateIV();
        AESUtil.encryptFile(inFile, outEncFile, aesKey, iv);

        byte[] encryptedAesKey =
        RSAUtil.encryptWithPublicKey(aesKey.getEncoded(), publicKey);
        String encryptedAesKeyB64 =
        Base64.getEncoder().encodeToString(encryptedAesKey);

        java.nio.file.Files.write(outEncKeyFile.toPath(),
        encryptedAesKeyB64.getBytes(StandardCharsets.UTF_8));

        return encryptedAesKeyB64;
    }

    public static void decryptFile(File inEncFile, File inEncKeyFile,
        java.security.PrivateKey privateKey, File outFile) throws Exception {
        String b64 = new String(java.nio.file.Files.readAllBytes(inEncKeyFile.toPath()),
        StandardCharsets.UTF_8);
        byte[] encryptedAesKey = Base64.getDecoder().decode(b64);
        byte[] aesKeyBytes = RSAUtil.decryptWithPrivateKey(encryptedAesKey,
        privateKey);
        javax.crypto.SecretKey aesKey =
        javax.crypto.spec.SecretKeySpec(aesKeyBytes, "AES");
        AESUtil.decryptFile(inEncFile, outFile, aesKey);
    }
}
```

Database.java

```
package com.cifrador.database;

import java.sql.*;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.Properties;

public class DatabaseHelper {
    private final String url;
    private final Properties props;

    public DatabaseHelper(String host, int port, String dbName, String user, String
password) {
        this.url =
String.format("jdbc:mysql://%s:%d/%s?useSSL=false&allowPublicKeyRetrieval=tru
e&serverTimezone=UTC", host, port, dbName);

        this.props = new Properties();
        props.setProperty("user", user);
        props.setProperty("password", password);
    }

    private Connection getConn() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return DriverManager.getConnection(url, props);
    }

    public void insertOperacion(String nombreArchivo, String
claveAesCifradaBase64, String usuario) throws SQLException {
        String sql = "INSERT INTO operaciones (nombre_archivo, fecha,
clave_aes_cifrada, usuario) VALUES (?, ?, ?, ?)";
        try (Connection c = getConn(); PreparedStatement ps =
c.prepareStatement(sql)) {
            ps.setString(1, nombreArchivo);
            ps.setTimestamp(2,
Timestamp.valueOf(LocalDateTime.now(ZoneId.of("UTC"))));
            ps.setString(3, claveAesCifradaBase64);
```

```
        ps.setString(4, usuario);  
        ps.executeUpdate();  
    }  
}
```