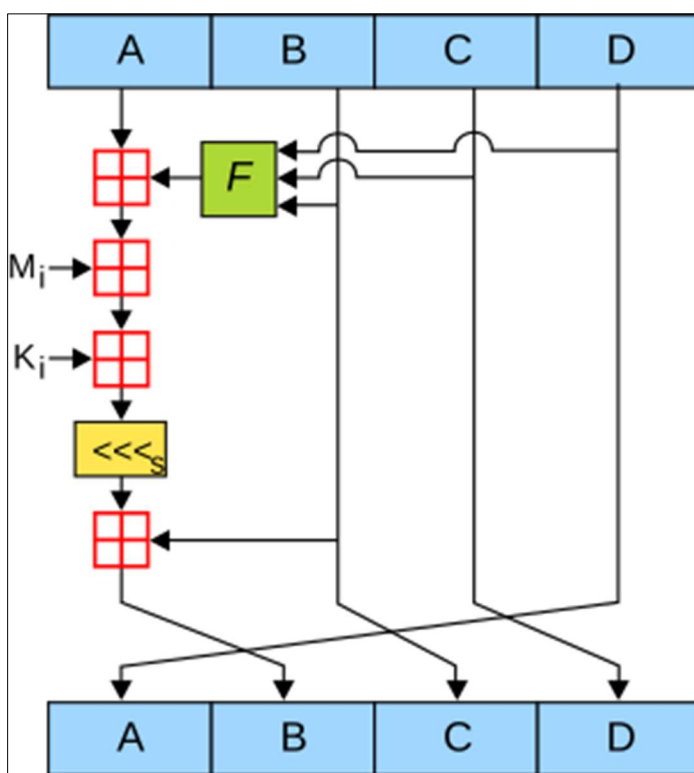


BÀI 5. HÀM BẮM VÀ ỨNG DỤNG

5.1 HÀM BẮM MD5

5.1.1 Giới thiệu:

MD5 (Message Digest Algorithm 5) là một trong những thuật toán băm (hash) được sử dụng rộng rãi để tạo ra một giá trị băm duy nhất (đôi khi được gọi là checksum hoặc hash value) từ một đầu vào dữ liệu. Nó được thiết kế để tạo ra một giá trị băm 128-bit, thường được biểu diễn bằng một chuỗi 32 ký tự hexa.



Tuy nhiên, cần lưu ý rằng MD5 đã bị các vấn đề về bảo mật phát hiện từ năm 1996 và các tấn công ngày càng phát triển dẫn đến việc MD5 không còn an toàn cho các ứng dụng bảo mật cao. Cách thức hoạt động của MD5:

- Băm dữ liệu: Đầu vào (dữ liệu cần băm) được chia thành các khối 512-bit và được xử lý theo từng khối.

- Thêm bit padding: Để đảm bảo rằng đầu vào có độ dài phù hợp, bit padding được thêm vào cuối dữ liệu.
- Giai đoạn nén: Mỗi khối dữ liệu được xử lý qua một loạt các vòng nén sử dụng hàm luân phiên (permutation function), hàm non-linear (non-linear function) và các phép lấy modulo.
- Tạo giá trị băm: Cuối cùng, sau khi tất cả các khối đã được xử lý, giá trị băm 128-bit được tạo ra từ các giá trị trung gian được tính toán trong quá trình nén.

5.1.2 Bài tập thực hành:

- Tạo package “week_05”. Trong package “week_05” tạo class “MD5Util.java”:

```
package week_05;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5Util {

    public static String md5(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(input.getBytes());
            byte[] digest = md.digest();
            BigInteger bigInt = new BigInteger(1, digest);
            String md5Hex = bigInt.toString(16);
            while (md5Hex.length() < 32) {
                md5Hex = "0" + md5Hex;
            }
            return md5Hex;
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

- Tạo JFrame Form “frm_MD5.java”:

The image shows a Java Swing window titled "MD5 Hash Demo". It contains two text input fields. The first field is labeled "Plaintext:" and is currently empty. The second field is labeled "Hash:" and is also empty. Below the "Hash:" field is a button labeled "MD5 Hash".

- Phương thức xử lý sự kiện nhấn nút “MD5 Hash”:

```
private void btn_encryptActionPerformed(java.awt.event.ActionEvent evt) {
    String input = txt_plaintext.getText();
    String md5Hash = MD5Util.md5(input);
    txt_hash.setText(md5Hash);
}
```

- Chạy form MD5 Hash Demo:

The image shows the same "MD5 Hash Demo" window, but now it is populated with data. The "Plaintext:" field contains the text "Nguyen Trong Minh Hong Phuoc". The "Hash:" field contains the MD5 hash "64da5e96b2a0bd95e04016667f3d044". The "MD5 Hash" button is still present at the bottom.

5.2 HÀM BẢM SHA

5.2.1 Giới thiệu:

Ban đầu, MD5 được sử dụng rộng rãi trong việc bảo mật mật khẩu và xác thực, cũng như trong việc kiểm tra tính toàn vẹn của các file. Tuy nhiên, do các vấn đề bảo mật

đã được phát hiện, nên các thuật toán khác như SHA-256, SHA-3 đã được khuyến nghị thay thế MD5 trong các ứng dụng yêu cầu mức độ bảo mật cao hơn.

SHA-256 là một trong các thuật toán băm thuộc họ SHA-2 (Secure Hash Algorithm 2), được thiết kế để tạo ra một giá trị băm có độ dài 256-bit từ một đầu vào dữ liệu bất kỳ, đảm bảo tính duy nhất và khó khăn để trở lại dữ liệu ban đầu.

Được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA), SHA-256 đã trở thành một trong những thuật toán băm phổ biến nhất và được sử dụng rộng rãi trong các ứng dụng bảo mật thông tin. Cách thức hoạt động của SHA-256:

- **Chuẩn bị đầu vào:** Đầu vào dữ liệu được chia thành các khối 512-bit.
- **Thêm bit padding:** Một số bit được thêm vào cuối dữ liệu để đảm bảo rằng kích thước cuối cùng của dữ liệu là 512-bit, theo định dạng của thuật toán.
- **Giai đoạn nén:** Các khối dữ liệu được xử lý qua một loạt các vòng nén sử dụng hàm luân phiên (permutation function), hàm non-linear (non-linear function), và các phép lấy modulo, tương tự như cách hoạt động của các thuật toán băm khác.

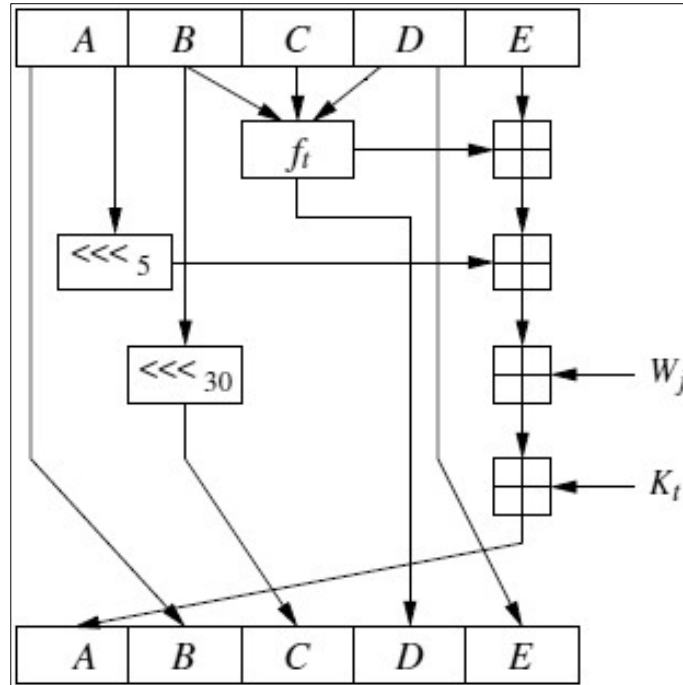
Tạo giá trị băm: Cuối cùng, sau khi tất cả các khối đã được xử lý, giá trị băm 256-bit được tạo ra từ các giá trị trung gian được tính toán trong quá trình nén.

SHA-3 (Secure Hash Algorithm 3) là một thuật toán băm được thiết kế để tạo ra các giá trị băm từ các đầu vào dữ liệu bất kỳ. Nó là một trong những tiêu chuẩn bảo mật được công nhận quốc tế, được phát triển bởi hội đồng tiêu chuẩn hóa Quốc tế (NIST - National Institute of Standards and Technology) và được công bố vào năm 2015.

Thuật toán SHA-3 không chỉ đơn thuần là một phiên bản nâng cấp của SHA-2, mà là một thuật toán băm hoàn toàn mới được xây dựng từ các cơ sở thiết kế hoàn toàn khác biệt:

- **Kiến trúc hoàn toàn khác biệt:** SHA-3 không chia sẻ cùng kiến trúc hoặc cơ sở với SHA-2.
- **Khả năng chống tấn công:** SHA-3 được thiết kế để chống lại các phương pháp tấn công phổ biến hiện đại như collision attacks (tấn công va chạm).
- **Độ dài băm linh hoạt:** SHA-3 có thể tạo ra các giá trị băm với độ dài khác nhau, không giống như SHA-2 chỉ tạo ra các giá trị băm với độ dài cố định.

- **Tính hiệu quả và hiệu suất:** SHA-3 có khả năng chạy trên nhiều nền tảng khác nhau với tốc độ cao và hiệu suất tốt.



5.2.2 Bài tập thực hành:

- Tạo class "SHAUtil.java":

```

package week_05;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHAUtil {

    public static String sha1(String input)
        throws NoSuchAlgorithmException {
        return hashString(input, "SHA-1");
    }

    public static String sha256(String input)
        throws NoSuchAlgorithmException {
        return hashString(input, "SHA-256");
    }

    public static String sha512(String input)
        throws NoSuchAlgorithmException {
        return hashString(input, "SHA-512");
    }

    private static String hashString(String input, String algorithm)
        throws NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance(algorithm);
        byte[] hashedBytes = md.digest(input.getBytes());
        StringBuilder sb = new StringBuilder();
        for (byte b : hashedBytes) {
            sb.append(String.format("%02x", b));
        }
        return sb.toString();
    }
}

```

- Tạo JFrame Form "frm_SHA.java":

SHA Hash Demo

Plaintext:

SHA-1:

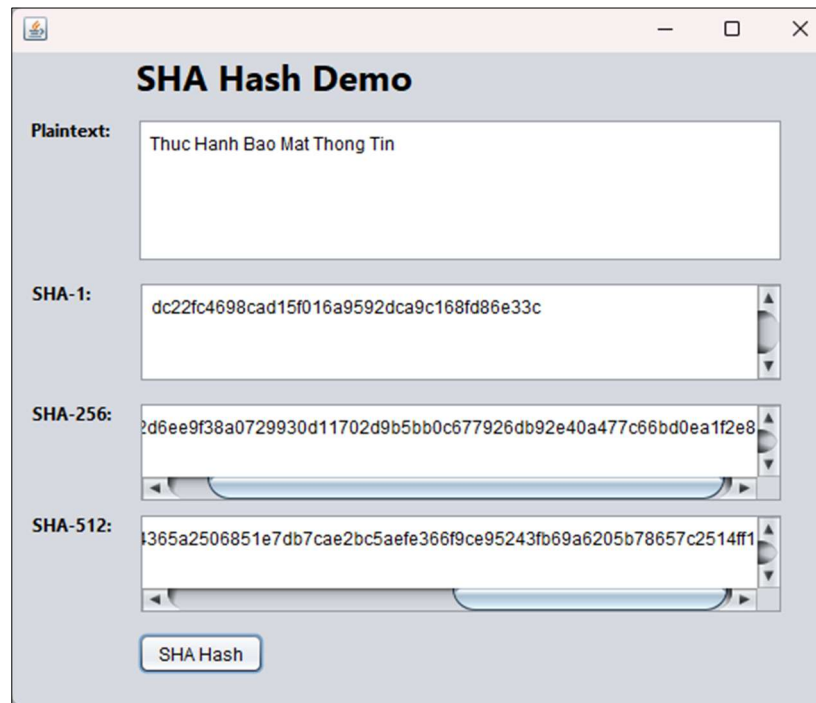
SHA-256:

SHA-512:

- Phương thức xử lý sự kiện khi nhấn nút “SHA Hash”:

```
private void btn_encrypt1ActionPerformed(java.awt.event.ActionEvent evt) {
    String plaintext = txt_plaintext.getText();
    if (plaintext.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Please enter plaintext.");
        return;
    }
    try {
        // Calculate SHA-1 hash
        String sha1Hash = SHAUtil.sha1(plaintext);
        txt_hash.setText(sha1Hash);
        // Calculate SHA-256 hash
        String sha256Hash = SHAUtil.sha256(plaintext);
        txt_hash1.setText(sha256Hash);
        // Calculate SHA-512 hash
        String sha512Hash = SHAUtil.sha512(plaintext);
        txt_hash2.setText(sha512Hash);
    } catch (NoSuchAlgorithmException ex) {
        JOptionPane.showMessageDialog(this,
            "Error calculating SHA hash: " + ex.getMessage());
    }
}
```

- Chạy form SHA Hash Demo:



5.3 KIỂM TRA TÍNH TOÀN VỆ CỦA TẬP TIN

5.3.1 Giới thiệu:

Kiểm tra tính toàn vẹn của tệp tin thông qua hàm băm là một kỹ thuật phổ biến trong lập trình và an ninh mạng. Hàm băm (hash function) là một hàm số toán học có khả năng chuyển đổi dữ liệu đầu vào (tệp tin, chuỗi dữ liệu) thành một giá trị băm (hash value) cố định có độ dài cố định, được gọi là băm (hash). Băm có những đặc điểm sau:

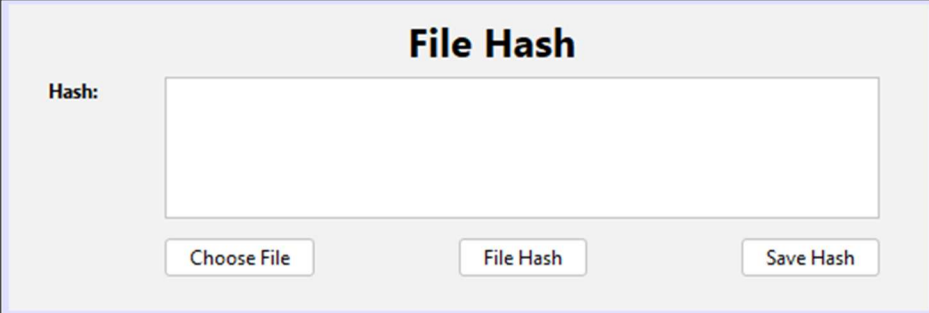
- Không thể đảo ngược: Không thể phục hồi dữ liệu ban đầu từ giá trị băm.
- Độ dài cố định: Dù dữ liệu đầu vào có lớn hay nhỏ thế nào đi nữa, kết quả băm luôn có độ dài cố định.
- Độc lập: Một sự thay đổi nhỏ trong dữ liệu đầu vào sẽ gây ra một sự thay đổi lớn và khó đoán trong giá trị băm.
- Độ phân tán: Các giá trị băm khác nhau phân tán đều trong không gian giá trị băm.

Cơ chế thực hiện kiểm tra tính toàn vẹn của tệp tin bằng hàm băm:

- Tạo giá trị băm (hash value): Đầu tiên, tính toán giá trị băm của tệp tin ban đầu bằng cách áp dụng hàm băm lên nội dung của tệp tin. Đây là quá trình đầu tiên khi tệp tin được tạo ra hoặc khi kiểm tra tính toàn vẹn của tệp tin.
- Lưu trữ giá trị băm: Lưu giá trị băm này lại, chẳng hạn trong một file riêng hoặc trong cơ sở dữ liệu, nhằm cho việc so sánh sau này.
- Kiểm tra tính toàn vẹn: Khi cần kiểm tra tính toàn vẹn của tệp tin, lại tính toán giá trị băm mới từ nội dung hiện tại của tệp tin. Sau đó, so sánh giá trị băm này với giá trị băm đã lưu trữ.
- Xử lý kết quả: Nếu giá trị băm tính được từ tệp tin hiện tại giống với giá trị băm đã lưu trữ, điều này cho thấy tệp tin không bị thay đổi (tức là tính toàn vẹn của tệp tin được bảo đảm). Ngược lại, nếu giá trị băm khác nhau, có sự thay đổi trong nội dung của tệp tin.

5.3.2 Bài tập thực hành:

- Tạo JFrame Form “frm_FileHash.java”:



- Thêm các gói và khởi tạo:

```

package week_05;

import java.io.BufferedWriter;
import javax.swing.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class frm_FileHash extends JFrame {

    private File selectedFile;

    public frm_FileHash() {
        initComponents();
    }

```

– Nút chọn file:

```

private void btn_choose_fileActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        selectedFile = fileChooser.getSelectedFile();
    }
}

```

– Nút thực hiện hàm băm:

```

private void btn_hashActionPerformed(java.awt.event.ActionEvent evt) {
    if (selectedFile == null) {
        JOptionPane.showMessageDialog(this,
            "Please choose a file first.", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try {
        byte[] fileBytes = readFile(selectedFile);
        String hash = calculateHash(fileBytes, "SHA-256");
        txt_hash.setText(hash);
    } catch (IOException | NoSuchAlgorithmException e) {
        JOptionPane.showMessageDialog(this,
            "Error calculating hash: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

– Nút lưu giá trị hàm băm vào file:

```
private void btn_saveActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser fileChooser = new JFileChooser();
    int option = fileChooser.showSaveDialog(this);
    if (option == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        String content = txt_hash.getText();
        if (!file.getName().toLowerCase().endsWith(".txt")) {
            file = new File(file.getParentFile(), file.getName() + ".txt");
        }
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
            writer.write(content);
            JOptionPane.showMessageDialog(this, "File saved successfully.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "Error saving file: "
                + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

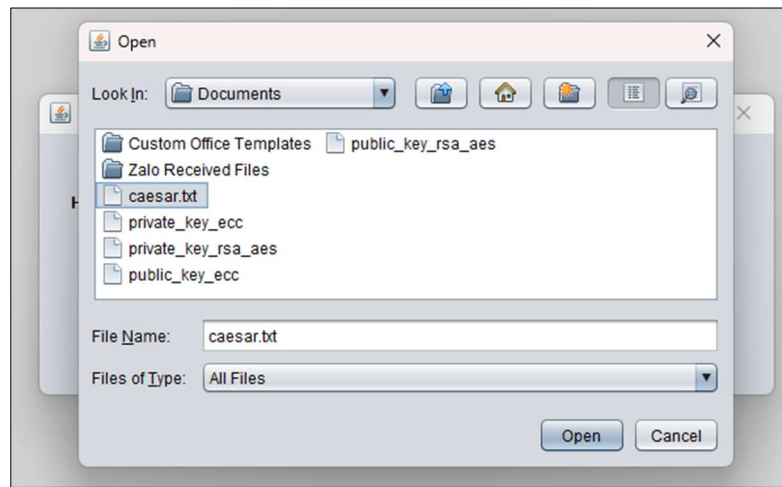
- Phương thức đọc file:

```
private byte[] readFile(File file) throws IOException {
    try (FileInputStream fis = new FileInputStream(file)) {
        byte[] buffer = new byte[1024];
        int bytesRead;
        StringBuilder sb = new StringBuilder();
        while ((bytesRead = fis.read(buffer)) != -1) {
            sb.append(new String(buffer, 0, bytesRead));
        }
        return sb.toString().getBytes();
    }
}
```

- Phương thức tính giá trị hàm băm:

```
private String calculateHash(byte[] input, String algorithm)
    throws NoSuchAlgorithmException {
    MessageDigest md = MessageDigest.getInstance(algorithm);
    byte[] hashBytes = md.digest(input);
    StringBuilder sb = new StringBuilder();
    for (byte b : hashBytes) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}
```

- Chạy form và kiểm tra:



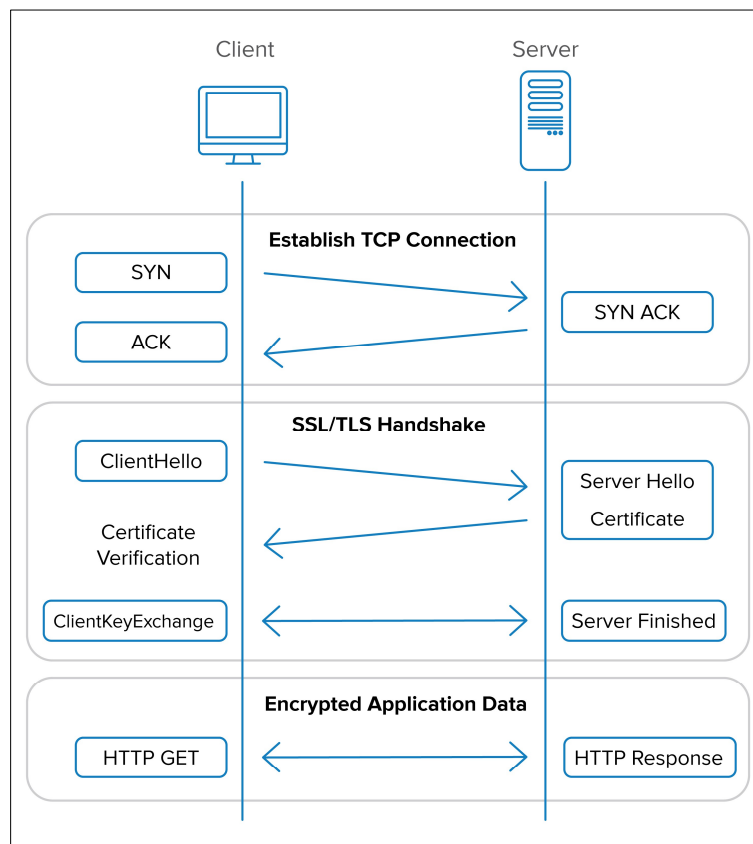
5.4 GIAO THỨC SSL VỚI SHA-256 VÀ RSA

5.4.1 Giới thiệu:

SSL (Secure Sockets Layer) là một công nghệ mật mã hóa dữ liệu được sử dụng để bảo vệ thông tin truyền tải qua internet. SSL đã được phát triển để cung cấp một lớp an ninh bổ sung cho việc truyền thông qua mạng, đặc biệt là trong việc bảo vệ giao tiếp giữa máy khách và máy chủ trong các ứng dụng web.

Cơ chế hoạt động chính của SSL là mật mã hóa dữ liệu khi chúng được truyền qua internet. Nó sử dụng một phương thức mã hóa đối xứng, thường là RSA, để mã hóa dữ liệu trước khi nó được gửi đi và giải mã nó khi dữ liệu đến đích. Quá trình mã hóa này giúp ngăn chặn kẻ tấn công từ việc đọc hoặc sửa đổi thông tin truyền tải.

SSL đã được nâng cấp và phát triển thành TLS (Transport Layer Security), phiên bản tiến bộ hơn của SSL. TLS thường được sử dụng rộng rãi hơn trong ngành công nghiệp ngày nay. Khi chúng truy cập một trang web có giao thức "https://" (với "s" là viết tắt của "secure"), đó chính là dấu hiệu cho thấy trang web sử dụng SSL hoặc TLS để bảo vệ giao tiếp giữa máy khách và máy chủ.



Cơ chế hoạt động của SSL (và TLS) bao gồm các bước chính sau:

- **Bắt đầu phiên kết nối (Handshake):** Khi một máy khách kết nối đến một máy chủ thông qua giao thức bảo mật SSL, quá trình bắt đầu bằng một bước gọi là "handshake".
- **Xác thực (Authentication):** Máy chủ gửi chứng chỉ số công khai của mình đến máy khách để xác minh danh tính của mình. Máy khách kiểm tra chứng chỉ để đảm bảo rằng máy chủ được xác thực và có thể tin cậy.
- **Trao đổi khóa (Key Exchange):** Máy khách và máy chủ sử dụng một loạt các thuật toán để thống nhất và trao đổi thông tin về khóa mã hóa.
- **Mã hóa và Giải mã (Encryption and Decryption):** Sau khi các thông tin khóa được trao đổi, máy khách và máy chủ sử dụng thông tin này để mã hóa và giải mã dữ liệu được truyền qua kết nối.
- **Kiểm tra Tính toàn vẹn (Integrity Check):** SSL cũng cung cấp kiểm tra tính toàn vẹn dữ liệu. Mỗi gói tin được gửi đi đều được đính kèm mã xác thực để ngăn chặn việc dữ liệu bị sửa đổi khi truyền qua mạng.

5.4.2 Bài tập thực hành:

- Tạo class "SSLUtil.java":

```
package week_05;

import org.bouncycastle.asn1.x500.X500Name;
import org.bouncycastle.cert.X509v3CertificateBuilder;
import org.bouncycastle.cert.jcajce.JcaX509CertificateConverter;
import org.bouncycastle.cert.jcajce.JcaX509v3CertificateBuilder;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.operator.ContentSigner;
import org.bouncycastle.operator.jcajce.JcaContentSignerBuilder;

import java.math.BigInteger;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.Security;
import java.security.cert.X509Certificate;
import java.util.Date;
```

```
public class SSLUtil {

    static {
        Security.addProvider(new BouncyCastleProvider());
    }

    public static KeyPair generateRSAKeyPair() throws Exception {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        keyPairGenerator.initialize(2048);
        return keyPairGenerator.generateKeyPair();
    }
}
```



```

public static X509Certificate
generateSelfSignedCertificate(KeyPair keyPair, String dn) throws Exception {
    long now = System.currentTimeMillis();
    Date startDate = new Date(now);

    X500Name dnName = new X500Name(dn);
    BigInteger certSerialNumber = new BigInteger(Long.toString(now));
    Date endDate = new Date(now + 365L * 24L * 60L * 60L * 1000L);

    ContentSigner contentSigner =
        new JcaContentSignerBuilder("SHA256WithRSA").build(keyPair.getPrivate());
    X509v3CertificateBuilder certificateBuilder = new JcaX509v3CertificateBuilder(
        dnName, certSerialNumber, startDate, endDate, dnName, keyPair.getPublic());

    return new JcaX509CertificateConverter()
        .setProvider("BC").getCertificate(certificateBuilder.build(contentSigner));
}

public static String convertToPEM(X509Certificate certificate) throws Exception {
    StringBuilder pem = new StringBuilder();
    pem.append("-----BEGIN CERTIFICATE-----\n");
    pem.append(new String(
        java.util.Base64.getMimeEncoder().encode(certificate.getEncoded())));
    pem.append("\n-----END CERTIFICATE-----");
    return pem.toString();
}
}

```

- Tạo JFrame Form "frm_SSL.java":

- Thêm các gói cần thiết và khởi tạo:

```

package week_05;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.security.KeyPair;
import java.security.cert.X509Certificate;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class frm_SSL extends javax.swing.JFrame {

    private X509Certificate generatedCertificate;

    public frm_SSL() {
        initComponents();
    }

```

- Sự kiện generate:

```

private void btn_generateActionPerformed(java.awt.event.ActionEvent evt) {
    generateCertificate();
}

```

- Phương thức tạo chứng chỉ:

```

private void generateCertificate() {
    try {
        String dn = "CN=" + txt_CN.getText() + ", O=" + txt_O.getText()
            + ", OU=" + txt_OU.getText() + ", L=" + txt_L.getText()
            + ", ST=" + txt_ST.getText() + ", C=" + txt_C.getText();
        KeyPair keyPair = SSLUtil.generateRSAKeyPair();
        generatedCertificate = SSLUtil.generateSelfSignedCertificate(keyPair, dn);
        txt_certificate.setText(generatedCertificate.toString());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this,
            "Error generating certificate: " + e.getMessage(),
            "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

- Nút lưu chứng chỉ:


```

private void btn_saveActionPerformed(java.awt.event.ActionEvent evt) {
    if (generatedCertificate == null) {
        JOptionPane.showMessageDialog(this,
            "No certificate to save. Please generate a certificate first.",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Certificate as PEM");
    int userSelection = fileChooser.showSaveDialog(this);
    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();
        if (!fileToSave.getAbsolutePath().endsWith(".pem")) {
            fileToSave = new File(fileToSave + ".pem");
        }
        try (FileWriter writer = new FileWriter(fileToSave)) {
            String pemCertificate = SSLUtil.convertToPEM(generatedCertificate);
            writer.write(pemCertificate);
            JOptionPane.showMessageDialog(this, "Certificate saved successfully.",
                "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                "Error saving certificate: " + e.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            Logger.getLogger(frm_SSL.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
}

```

- Kiểm tra ứng dụng:

SSL Certificate Generater (SHA-256 & RSA)

Common Name (CN): Cong Ty Co Phan CNTT ABC

Organization (O): ABC Company

Organizational Unit (OU): ABC Company

Locality (L): Thu Duc City

State (ST): Ho Chi Minh City

Country (C): Viet Nam

SSL Certificate:

```

[0] Version: 3
SerialNumber: 1719319666574
IssuerDN: CN=Cong Ty Co Phan CNTT ABC,O=ABC Company,
Start Date: Tue Jun 25 19:47:46 ICT 2024
Final Date: Wed Jun 25 19:47:46 ICT 2025
SubjectDN: CN=Cong Ty Co Phan CNTT ABC,O=ABC Company,
Public Key: RSA Public Key [2c:45:73:27:73:cd:e5:58:cf:0b:26:92:
modulus: d69cdf46ea06bb027e7d49abd915de76a6fd5a3b0e442

```

Generate Certificate Save Certificate



```

1 -----BEGIN CERTIFICATE-----
2 MIIDqDCCApCgAwIBAgIGA ZBPcB+OMA0GCSqGSIb3DQEBCwUAMIGUMSEwHwYDVQQDDbDb25nIFR5
3 IENvIFBoYW4gQ05UVCBBQkMxFDASBgNVBAoMC0FCQyBDb21wYW55MRQwEgYDVQQLDAtBQkMgQ29t
4 cGFueTEVMBMGA1UEBwwMVGh1IER1YyBdaXR5MRkwFwYDVQQIDBBIbyBdaGkgTWluaCBdaXR5MREw
5 DwYDVQQGEWhWaWV0IE5hbTAeFw0yNDA2MjUxMjQ3NDZaFw0yNTA2MjUxMjQ3NDZaMIGUMSEwHwYD
6 VQQDDbDb25nIFR5IENvIFBoYW4gQ05UVCBBQkMxFDASBgNVBAoMC0FCQyBDb21wYW55MRQwEgYD
7 VQQLDAtBQkMgQ29tcGFueTEVMBMGA1UEBwwMVGh1IER1YyBdaXR5MRkwFwYDVQQIDBBIbyBdaGkg
8 TWluaCBdaXR5MREwDwYDVQQGEWhWaWV0IE5hbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
9 ggEBANac30bqBrsCfn1Jq9kV3nam/Vo7DkQhR4driKSk+5mTuMMkWfh/V5sKO2wDI9qgHb+zpw9/
10 A7wDYtmhqNI fKNACfdLdvEivs63HK6E2ssxAW3j+OsQITp2zzO4Tw3Q/COtgqaVsHu3cbA/ox tZR
11 mQQYcOX7d8JhD6hVbZJFcunCSryq8Vf1eUJIG5qIxchckoLIvEpBa64YHRahpMuJVE xRy+23augE
12 aICWYCh1fm2Pb1CGg5Hr59mkmzWywvu+p9faWYbZ0rjywIKQrDBkv3is3s9488o23xJWWUPXYBWW
13 N6RU p598/Q3UdDw5QIL9plfr+Dtpec6CL1t3RGFjZOkCAwEAATANBgkqhkiG9w0BAQsFAAOCAQEA
14 lQeog54jvg9v0dS3sZfrQVv8grHWqnuEbQMPsss2jtcajHdQBHt9ITC+fv17i1lnzgmK59iRKVSi
15 hgd/LjWa3v6GU5bxdGJxeAwhV14MQpzEI9iPrjHPO2t4vPj scbumEz605NPwbKy7Tb0pdWELc0pm
16 GTLMBn9fQM+Kx4wKVe bPIsRznAJan8qZCVS1W3m6pjB9arWelmjBEkn1hIDI faBW i3wEYlMhD/S
17 DT2G2JnVA8tsZuaB3vdKVl2cVH3kB2QM5hb97uBIS0pOoes5/Pd3Zlvxqh qLyRBPI6r62lVT55ZK
18 T0T6zGjP29S29EKGcxilGnJcwBpnrM2kqXRQgQ==
19 -----END CERTIFICATE-----

```

5.5 BÀI TẬP NÂNG CAO

Tìm hiểu và viết chương trình thực hiện hàm băm Bcrypt. Bcrypt có thể được sử dụng để bảo vệ mật khẩu và các dữ liệu quan trọng bằng cách tạo ra một chuỗi băm mạnh mẽ, khó khăn cho các cuộc tấn công vét cạn (brute-force) hoặc từ điển. Sử dụng thư viện Java: jBCrypt.

