БЕЗБЕДНОСТ НА КОМПЈУТЕРСКИ СИСТЕМИ

ЛАБОРАТОРИСКА ВЕЖБА 1

Фисник Лимани, 151027

Во следните слики ќе ги прикажеме составните делови на кодот на првата лабораториска вежба и ќе го видиме/коментираме што им е нивната функција.

1. Класата **`FrameHeader`**

```java
public class FrameHeader {
    private byte[] sourceMAC;
    private byte[] destinationMAC;

    public FrameHeader(byte[] sourceMAC, byte[] destinationMAC){
        this.sourceMAC = sourceMAC;
        this.destinationMAC = destinationMAC;
    }

    public byte[] getSourceMAC() {
        return sourceMAC;
    }

    public byte[] getDestinationMAC() {
        return destinationMAC;
    }
}
```

2. Класата **`ClearTextFrame`**

```java
public class ClearTextFrame {
    private FrameHeader frameHeader;
    private byte[] data;

    public ClearTextFrame(FrameHeader frameHeader, byte[] data) {
        this.frameHeader = frameHeader;
        this.data = data;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();

        sb.append("Source MAC: ");
        sb.append(new String(frameHeader.getSourceMAC()) + "\n");
        sb.append("Destination MAC: ");
        sb.append(new String(frameHeader.getDestinationMAC()) + "\n");
        sb.append("Payload: ");
        sb.append(new String(data) + "\n");

        return sb.toString();
    }
    // ...
```

“// …” на крај на сликата значи дека оваа класа содржи уште некои функции кои ќе бидат прикажани подолу.

3. Класата **`EncryptedFrame`**

```java
public class EncryptedFrame {
    private byte[] sourceMAC;
    private byte[] destinationMAC;
    private byte[] mic;
    private byte[] encryptedData;

    public EncryptedFrame(byte[] sourceMAC, byte[] destinationMAC, byte[] encryptedData, byte[] mic) {
        this.sourceMAC = sourceMAC;
        this.destinationMAC = destinationMAC;
        this.mic = mic;
        this.encryptedData = encryptedData;
    }

    @Override
    public String toString(){
        StringBuilder sb = new StringBuilder();

        sb.append("Source MAC: ");
        sb.append(new String(sourceMAC) + "\n");
        sb.append("Destination MAC: ");
        sb.append(new String(destinationMAC) + "\n");
        sb.append("Payload: ");
        sb.append(Base64.getEncoder().encodeToString(encryptedData) + "\n");
        sb.append("MIC: ");
        sb.append(Base64.getEncoder().encodeToString(mic));

        return sb.toString();
    }
    // ...
```

4. Класата **`CCMProtocol`**

```
public class CCMProtocol {
    private SecretKey secretKey;
    private Cipher micCipher;
    private Cipher encryptionCipher;

    public CCMProtocol() throws NoSuchAlgorithmException, NoSuchPaddingException {
        // GENERATE KEY
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        SecureRandom secureRandom = new SecureRandom();
        int keyBitSize = 128;
        keyGenerator.init(keyBitSize, secureRandom);
        this.secretKey = keyGenerator.generateKey();
        // AES_CBC mode
        this.micCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        // AES_CTR mode
        this.encryptionCipher = Cipher.getInstance("AES/CTR/PKCS5Padding");
    }
    // ...
```

- `CCMProtocol` класата го симулира CCM протоколот.
  Во оваа класа го изгенерираме:
  - Клучот (Key-то)
  - AES/CBC шифрувачот
  - AES/CTR шифрувачот

5. Во класата `ClearTextFrame` ги имаме уште следните функции:
   - **`encryptFrame(Cipher micCipher, Cipher encryptionCipher)`**
     - Оваа функција е централната функција во која се врши енкрипција на рамката
     - Прво се прави пресметување на MIC со посебна функција
     - Потоа се шифрира пораката

```
public EncryptedFrame encryptFrame(Cipher micCipher, Cipher encryptionCipher) throws BadPaddingException, IllegalBlockSizeException {
    byte[] mic = calculateMIC(micCipher);
    byte[] encryptedBytes = encryptData(encryptionCipher, mic);

    EncryptedFrame encryptedFrame = new EncryptedFrame(
            frameHeader.getSourceMAC(),
            frameHeader.getDestinationMAC(),
            Arrays.copyOfRange(encryptedBytes, from: 16, encryptedBytes.length),
            Arrays.copyOfRange(encryptedBytes, from: 0, to: 8)
    );
    return encryptedFrame;
}
```

- **`calculateMIC(Cipher cipher)`**
    - Функција за пресметување на MIC (Message Integrity Check)

```
private byte[] calculateMIC(Cipher cipher) throws BadPaddingException, IllegalBlockSizeException {
    byte[] dataForCalculatingMIC = prepareDataForCalculatingMIC();
    byte[] cbc_encrypted = cipher.doFinal(dataForCalculatingMIC);
    // MIC is the first 8 bytes in the last 16 bytes
    byte[] mic = new byte[8];

    for(int i = 0; i < mic.length; ++i){
        mic[i] = cbc_encrypted[cbc_encrypted.length - 16 + i];
    }
    return mic;
}
```

- **`prepareDataForCalculatingMIC()`**
    - Функција каде ги спремаме податоците за пресметување на MIC-от
    - Тука вршиме подредување на потребните податоци:
        - првите бајти ни се од изворниот MAC,
        - следните бајти ни се од дестинацискиот MAC,
        - и последните бајти ни се од пораката што сакаме да се испраќа

```
private byte[] prepareDataForCalculatingMIC(){
    byte[] sourceMAC = frameHeader.getSourceMAC();
    byte[] destinationMAC = frameHeader.getDestinationMAC();
    byte[] bytes = new byte[sourceMAC.length + destinationMAC.length + this.data.length];
    int i = 0;
    for(int j = 0; j < sourceMAC.length; ++j){
        bytes[i] = sourceMAC[j];
        i++;
    }
    for(int j = 0; j < destinationMAC.length; ++j){
        bytes[i] = destinationMAC[j];
        i++;
    }
    for(int j = 0; j < data.length; ++j){
        bytes[i] = data[j];
        i++;
    }
    return bytes;
}
```

- **`encryptData(Cipher cipher, byte[] mic)`**
    - Централна функција за шифрирање на пораката што се испраќа

```
private byte[] encryptData(Cipher cipher, byte[] mic) throws BadPaddingException, IllegalBlockSizeException {
    byte[] bytesToEncrypt = prepareDataToEncrypt(mic);
    byte[] encryptedBytes = cipher.doFinal(bytesToEncrypt);
    return encryptedBytes;
}
```

- **`prepareDataToEncrypt()`**
    - Функција за спремање на податоците за шифрирање
    - Првите 16 бајти ни се:
        - Првите 8 од MIC-от
        - Следните 8 може да бидат било какви бајти бидејќи ни се потребни само за да имаме еден блок од 128 битови (кои потоа ќе се игнорираат)
    - Следните бајти ни се од пораката што сакаме да се испраќа

```
private byte[] prepareDataToEncrypt(byte[] mic){
    byte[] bytesToEncrypt = new byte[mic.length + 8 + data.length];
    for(int i = 0; i < mic.length; ++i){
        bytesToEncrypt[i] = mic[i];
    }
    for(int i = 0; i < data.length; ++i){
        bytesToEncrypt[i + mic.length + 8] = data[i];
    }
    return  bytesToEncrypt;
}
```

    - Го ставиме MICот како прв блок во низата од блокови што ќе се шифрираат бидејќи така ќе постигнеме MICот да се шифрира со CTR0, а потоа другите блокови кој ќе се креираат од пораката ќе се шифрират со: CTR1, CTR2, … CTRm.

6. Во класата **EncryptedFrame** ги имаме уште следните функции:
   - **`decryptFrame(Cipher micCipher, Cipher decryptCipher)`**
     - Прво го дешифрираме рамката
     - Потоа го провериме MIC-от
       - Ако проверката на MIC-от поминува, тогаш се враќа инстанца од класата `ClearTextFrame`
       - Ако не, тогаш се фрли IllegalStateException исклучок

```
public ClearTextFrame decryptFrame(Cipher micCipher, Cipher decryptionCipher) throws BadPaddingException, IllegalBlockSizeException {
    byte[] decryptedBytes = decryptBytes(decryptionCipher);

    byte[] mic = Arrays.copyOfRange(decryptedBytes,  from: 0,  to: 8);
    byte[] data = Arrays.copyOfRange(decryptedBytes,  from: 16, decryptedBytes.length);

    if(verifyMIC(data, mic, micCipher)){
        ClearTextFrame clearTextFrame = new ClearTextFrame(new FrameHeader(sourceMAC, destinationMAC), data);
        return clearTextFrame;
    }else{
        throw new IllegalStateException();
    }
}
```

   - **`decryptBytes(Cipher decryptionCipher)`**
     - Со оваа функција се прави дешифрирање на рамката

```
private byte[] decryptBytes(Cipher decryptionCipher) throws BadPaddingException, IllegalBlockSizeException {
    byte[] cipherText = new byte[16 + encryptedData.length];
    for(int i = 0; i < mic.length; ++i){
        cipherText[i] = mic[i];
    }
    for(int i = 0; i < encryptedData.length; ++i){
        cipherText[16 + i] = encryptedData[i];
    }
    byte[] decryptedBytes = decryptionCipher.doFinal(cipherText);
    return decryptedBytes;
}
```

- **`verifyMIC(byte[] data, byte[] mic, Cipher cipher)`**
    - Со оваа функција се прави проверката за интегритетот на рамката

```
private boolean verifyMIC(byte[] data, byte[] mic, Cipher cipher) throws BadPaddingException, IllegalBlockSizeException {
    byte[] decryptedFrame = new byte[sourceMAC.length + destinationMAC.length + data.length];
    int i = 0;
    for(int j = 0; j < sourceMAC.length; ++j){
        decryptedFrame[i] = sourceMAC[j];
        i++;
    }
    for(int j = 0; j < destinationMAC.length; ++j){
        decryptedFrame[i] = destinationMAC[j];
        i++;
    }
    for(int j = 0; j < data.length; ++j){
        decryptedFrame[i] = data[j];
        i++;
    }
    byte[] cbc_encrypted = cipher.doFinal(decryptedFrame);
    byte[] micVerify = new byte[8];
    for(int j = 0; j < 8; ++j){
        micVerify[j] = cbc_encrypted[cbc_encrypted.length - 16 + j];
    }
    for(int j = 0; j < micVerify.length; ++j){
        if(mic[j] != micVerify[j]){
            return false;
        }
    }
    return true;
}
```

7. ДЕМО:

Вредностите со кои ќе ги тестираме нашите функции ќе бидат:

- Source MAC: **`E8:6A:23:A4:E8:51`**
- Destination MAC: **`34:40:10:2A:15:2A`**
- Data: **`ahhh … i'm tired of this lab :S:S:S`**

```
String sourceMAC = "E8:6A:23:A4:E8:51";
String destinationMAC = "34:40:10:2A:15:2A";
FrameHeader frameHeader = new FrameHeader(
        sourceMAC.getBytes( charsetName: "UTF-8"),
        destinationMAC.getBytes( charsetName: "UTF-8")
);
String data = "ahhh ... i'm tired of this lab :S:S:S";
ClearTextFrame clearTextFrame = new ClearTextFrame(frameHeader, data.getBytes( charsetName: "UTF-8"));
```

Main функцијата:

```
public static void main(String[] args) throws IOException {
    Security.addProvider(new BouncyCastleProvider());

    try{
        CCMProtocol CCMProtocol = new CCMProtocol();

        System.out.println("ENCRYPTION: ");
        EncryptedFrame encryptedFrame = CCMProtocol.encryptFrame(generateClearTextFrame());
        System.out.println(encryptedFrame);
        System.out.println();
        //        encryptedFrame.getSourceMAC()[3] = new Byte("0");
        //        encryptedFrame.getDestinationMAC()[3] = new Byte("0");
        //        encryptedFrame.getMic()[2] = new Byte("0");
        //        encryptedFrame.getEncryptedData()[5] = new Byte("0");
        ClearTextFrame decryptedFrame = CCMProtocol.decryptFrame(encryptedFrame);
        System.out.println("DECRYPTION: ");
        System.out.println(decryptedFrame);
    }
    catch(IllegalStateException | NoSuchAlgorithmException | NoSuchPaddingException e){
        System.out.println("IllegalStateException");
    }
}
```

Output-от кога шифрирањето и дешифрирањето се извршуваат успешно:

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
ENCRYPTION:
Source MAC: E8:6A:23:A4:E8:51
Destination MAC: 34:40:10:2A:15:2A
Payload: 9f1J1FrohGfPBymGnkicmB9y+56Voh4o5aYRf0AKvPpvIsBk9w==
MIC: 9FursfV6vhs=

DECRYPTION:
Source MAC: E8:6A:23:A4:E8:51
Destination MAC: 34:40:10:2A:15:2A
Payload: ahhh ... i'm tired of this lab :S:S:S
```

Output-от ако дешифрирањето не успее:

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
ENCRYPTION:
Source MAC: E8:6A:23:A4:E8:51
Destination MAC: 34:40:10:2A:15:2A
Payload: 8Y00VPkZfjcI9OiFyb+4+/YSxkGGmuRwvLrOIapuPuxeYLhc/w==
MIC: pEzGo+WNFyw=

IllegalStateException
```