



Master of Information Security

MIS4203 – Independent Studies in Information Security

Index Number – 16770217 | Reg. Number – 2016MIS021

Study Number – 09 - Cryptography and Java Security

January 09, 2019

University of Colombo School of Computing

Table of Contents

Study Number – 09 - Cryptography and Java Security 1

Problem..... 3

Approach 3

Conclusion 7

Problem

This study is about encryption and decryption in Java. Please download **is9.zip** file from LMS. In the folder you will find the files **encCTR.enc**, **encCCM.enc** and **encRSA.enc**, these are files that have been encrypted with AES in **CTR mode**, AES in **CCM mode** and **RSA**. You will also find the Java file **CryptoTool.java** which is a command line tool that can be used to encrypt files.

This tool uses the “Bouncy Castle” crypto libraries. These can be found in the bcprovjdk15on-151 jar file in the folder, and Java needs to be pointed to this.

So you can compile the code using:

```
javac -cp bcprov-jdk15on-151.jar CryptoTool.java
```

To run it using:

```
java -cp bcprov-jdk15on-151.jar:.. CryptoTool <mode> <key:128 bits in as hex> <inputFile> <outputFile>
```

The encCTR.enc file was created using the command:

```
java CryptoTool -encAESCTR 3eafda76cb8b015641cb946708675423  
plainText.txt encCTR.enc
```

The encCCM.enc file was created using the command:

```
java CryptoTool -encAESCCM 3eafda76cb8b015641cb946708675423  
plainText.txt encCCM.enc
```

RSA encryption must also includes the name of the keyStore:

```
java CryptoTool -encRSA <keyStore> <keyName> <inputFile> <outputFile>
```

Where keyStore is the file name of a key store and keyName is the name of the RSA key within the store. The tool will ask you for the password for the keystore (Bob is using ”bob”). The file myKeyStore contains an RSA key pair called mykey and is protected with the password: password.

The encRSA.enc file was created with the command:

```
java CryptoTool -encRSA myKeyStore mykey plainText.txt encRSA.enc
```

Approach

First of all need to import given java program and library to java IDE such as Eclipse or IntelliJ Idea and make the program compile.

Task 01

Complete the method decryptAESCTR(), so that the program can decrypt messages encrypted with AES in CTR mode.

Decrypt the encCTR.enc file.

```
private static void decryptAESCTR() {  
    try {  
        RandomAccessFile rawDataFromFile = new RandomAccessFile("encCTR.enc", "r");  
        byte[] cipherText = new byte[(int) rawDataFromFile.length()];  
        rawDataFromFile.read(cipherText);  
        rawDataFromFile.close();  
  
        SecretKeySpec secretKeySpec = new SecretKeySpec(hexStringToByteArray(hexKey), "AES");  
        Cipher cipher = Cipher.getInstance("AES/CTR/PKCS5PADDING");  
  
        byte iv[] = Arrays.copyOfRange(cipherText, 0, 16);  
        IvParameterSpec ivSpec = new IvParameterSpec(iv);  
  
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec);  
  
        byte[] c = Arrays.copyOfRange(cipherText, 16, cipherText.length);  
  
        String data = new String (cipher.doFinal(c));  
  
        writeContentToFile(data, "Q1_AESCTR_Decrypted_2016mis021.txt");  
    } catch (Exception e) {  
        System.out.println("Error while decrypting: " + e.toString());  
    }  
}
```

CryptoTool.java FileOutputStream.class Q1_AESCTR_Decrypted_2016mis021.txt

1 Well done. You have successfully decrypted a file encrypted with AES/CTR.
2 Submit this b70636b89763c217fb6632f86bec7c80

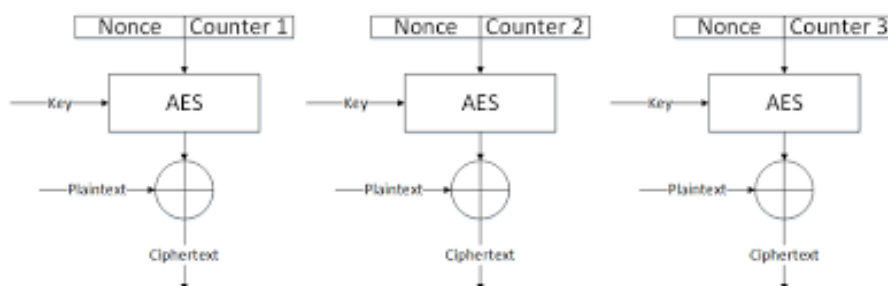
Task 02

CTR mode encryption does not authenticate the encrypted data, therefore it can be altered by the attacker. The plaintext of the message encrypted in the file ctr.enc is Pay Tom 1000 pounds, you do not have access to the encryption key.

Change the ciphertext so that the decrypted message would read Pay Bob 9999 pounds.

We strongly suggest that you make your own cipher texts with CryptoTool and experiment with them, and using the hex editor.

According to how CTR works is block wise encryption with same key diagram shown below.



For the encryption: Plain text will be XOR with key. So if we can guess/know the plain text value and XOR with cipher text, we can get the key. From that concept below is the calculation hex values for modified message. (value replace to Bob with Tom)

```
Q2 ×
70 82 E4 F2 9F C5 7B E4 86 01 60 8E 31 91 39 C1 14 D8 81

P a y      B o b      1 0 0 0      p o u n d s
P a y      T o m      9 9 9 9      p o u n d s

Bob 426f62  enc: 9fc57b  xor - random : ddaa19
Tom: 546f6d  random: ddaa19 --> 89c574

70 82 E4 F2 89 C5 74 E4 8E 08 69 87 31 91 39 C1 14 D8 81
```

Original:

```
Q2enc.enc
0 7082E4F2 9FC578E4 8601608E 319139C1 14D881 pÇ%Úú~{‰Ú `é1ë9; ŷÅ

Edited:
Q2enc.enc
0 7082E4F2 89C574E4 8E086987 319139C1 14D881 pÇ%Úá≈t‰é íá1ë9; ŷÅ
```

Decrypted output:

```
Problems Javadoc Declaration Console
<terminated> CTRModeAttack [Java Application] /Library/Java/Java
plain : Pay Bob 9999 pounds
```

Java code will be attached to the report:

Task 03

Complete the method `decryptAESCCM()`, so that the program can decrypt message encrypted with AES in CCM mode. Decrypt the `encCCM.enc` file. Also show editing some cipher texts and CCM mode detects the changes or not.

```
private static void decryptAESCCM() {
    try {
        // Open and read the input file
        // N.B. this program reads the whole file into memory, not good for large programs!
        RandomAccessFile rawDataFromFile = new RandomAccessFile("encCCM.enc", "r");
        byte[] cipherText = new byte[(int) rawDataFromFile.length()];
        rawDataFromFile.read(cipherText);
        rawDataFromFile.close();

        //Set up the AES key & cipher object in CCM mode
        SecretKeySpec secretKeySpec = new SecretKeySpec(hexStringToByteArray(hexKey), "AES");
        // Add a security provider that actually does provide CCM mode
        Security.insertProviderAt(new BouncyCastleProvider(), 1);
        Cipher cipher = Cipher.getInstance("AES/CCM/NoPadding","BC");

        byte iv[] = Arrays.copyOfRange(cipherText, 0, 10);
        IvParameterSpec ivSpec = new IvParameterSpec(iv);

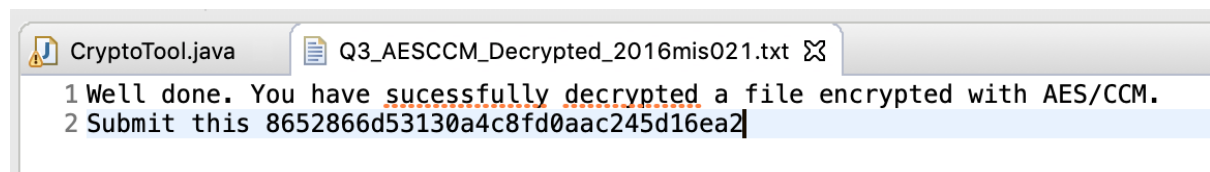
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec);

        byte[] c = Arrays.copyOfRange(cipherText, 10, cipherText.length);

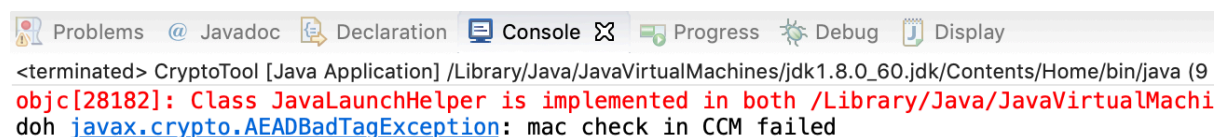
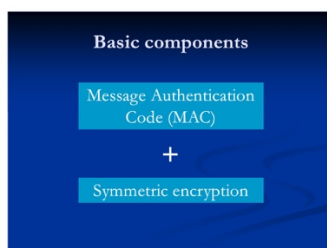
        String data = new String (cipher.doFinal(c));

        writeContentToFile(data, "Q3_AESCCM_Decrypted_2016mis021.txt");

    } catch (Exception e){
        System.out.println("doh "+e);
    }
}
```



Editing cipher text will fail the decryption process as it is CCM authenticating message content.



```
public class AEADBadTagException
    extends BadPaddingException
```

This exception is thrown when a Cipher operating in an AEAD mode (such as GCM/CCM) is unable to verify the supplied authentication tag.

Since:
1.7

See Also:
Serialized Form

Task 04

Complete the method decryptRSA(), so that the can decrypt message encrypted with RSA mode. Decrypt the encRSA.enc file..

```
private static void decryptRSA() {
    // System.out.println("RSA Decryption not yet supported");

    try {
        RandomAccessFile rawDataFromFile = new RandomAccessFile("encRSA.enc", "r");
        byte[] cipherText = new byte[(int) rawDataFromFile.length()];
        rawDataFromFile.read(cipherText);

        byte[] aesKey = Arrays.copyOfRange(cipherText, 0, 128);

        PrivateKey pkey = getPrivateKey(keyStore, keyName);
        Cipher rsaCipher = Cipher.getInstance("RSA");
        rsaCipher.init(Cipher.DECRYPT_MODE, pkey);

        byte[] encodedKey = rsaCipher.doFinal(aesKey);

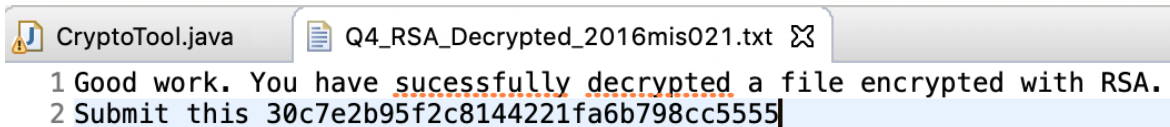
        Cipher aesCipher = Cipher.getInstance("AES");

        SecretKey skey = new SecretKeySpec(encodedKey, 0, encodedKey.length, "AES");
        aesCipher.init(Cipher.DECRYPT_MODE, skey);

        byte[] c = Arrays.copyOfRange(cipherText, 128, cipherText.length);
        String s = new String(aesCipher.doFinal(c));

        writeContentToFile(s, "Q4_RSA_Decrypted_2016mis021.txt");
        //System.out.println(s);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



```
1 Good work. You have sucessfully decrypted a file encrypted with RSA.
2 Submit this 30c7e2b95f2c8144221fa6b798cc5555]
```

Conclusion

1. AES CTR mode can be attacked by changing cipher text since it is not authenticating the message and it is not detecting by the algorithm.
2. AES CCM mode cannot be attacked by changing the cipher text as it is failing decryption by the algorithm when authenticating the message.
3. RSA mode will be used to protect symmetric key and share it between sender and recipient and message encryption will be done by the symmetric key.

***** End *****