

Task 2: Session Cipher

Due 30 Nov 2022 by 17:00 **Points** 0

Submitting a text entry box or a website url

Available 10 Nov 2022 at 17:00 - 30 Nov 2022 at 17:00

This assignment was locked 30 Nov 2022 at 17:00.

The next step is to use the session key you created in the previous assignment, and use it to create a *cipher* to encrypt and decrypt data. A cipher represents the parameters needed to specify encryption/decryption operations. For the purpose of encrypting a communication session using symmetric encryption, a cipher has five main components:

Parameter	Remark
Cryptographic algorithm	Use AES
Mode of operation	Use CTR (counter) mode
Padding algorithm	Use no padding
Secret key	All possible lengths allowed (128, 192, and 256 bits)
Initial Vector (IV)	Initial counter value. Same length as an AES block (128 bits).

The SessionCipher class

SessionCipher is a class that performs encryption and decryption on a stream of data. The data can come from a file, for instance or, as in the final project, from a network socket.

There are two constructors for SessionCipher. The first takes a SessionKey as parameter:

```
SessionCipher(SessionKey key)
```


With this constructor, the SessionCipher itself creates the IV.

The second constructor takes a SessionKey and an existing IV as parameters. The IV is represented as a byte array.

```
SessionCipherer(SessionKey key, byte[] ivbytes)
```

There are two "getter" methods to fetch the key/IV encryption parameters from a SessionEncrypter as byte arrays:

```
SessionKey[] getSessionKey()  
byte[] getIVBytes()
```

For the actual encryption, we use Java I/O Streams to communicate with the SessionCipher in the following way. Data to encrypt is sent to the SessionCipher via a [CipherOutputStream](https://docs.oracle.com/javase/8/docs/api/javax/crypto/CipherOutputStream.html) 

(<https://docs.oracle.com/javase/8/docs/api/javax/crypto/CipherOutputStream.html>),

object associated with the SessionCipher. The encrypted output from the SessionCipher goes to another [OutputStream](https://docs.oracle.com/javase/8/docs/api/java/io/OutputStream.html) 

(<https://docs.oracle.com/javase/8/docs/api/java/io/OutputStream.html>)_object, which can be a file, socket, etc. The following SessionCipher method sets up this arrangement, with a SessionCipher that is streaming data from a CipherOutputStream to an OutputStream:

```
CipherOutputStream openEncryptedOutputStream(OutputStream output)
```

So the idea is that after calling openEncryptedOutputStream, the caller can write data to the returned CipherOutputStream, and the data will then be encrypted and written to the output OutputStream. This may sound complicated, but hopefully it will be clearer if you look at the testing code (see below).

For decryption, there is a corresponding method to set up an InputStream.

```
CipherInputStream openDecryptedInputStream(InputStream input)
```

After calling `openDecryptedInputStream`, the caller can use the returned `CipherInputStream` to read plaintext data from it. Again, look at the testing code below.

Assignment

Implement the `SessionCipher` class in Java, with the methods and constructors described above.

To help you get started, there are skeleton files with all the required declarations in your personal repository on KTH GitHub.

Tip: study JCA carefully, read online tutorials and look at examples. If you do this right, The class will only be a few lines of codes.

Testing

The primary test is straightforward: take a piece of text and encrypt it into a ciphertext. Decrypt the cipher text, and check that the result matches. You find a JUnit 5 test program in your personal repository on KTH GitHub. It also servers as an example of the `SessionCipher` in action.

Submission

In order to complete your submission, you need to do two things:

- Commit your changes in git and push to your personal repository on KTH GitHub.
 - **Note:** the file should not contain any package declarations. If your IDE inserts package declarations for you, you need to remove them.
- Submit the URL of your personal KTH GitHub repository here, in this assignment, to notify us about the submission.

Important: We will not automatically check your personal KTH GitHub repository for updates. Whenever you make an update that you want us know about, you have to make a submission here. Submit the URL of your personal KTH GitHub repository, as described above.

