Directions

A. Due Date: 2 April 2025 at 11:55pm

B. The homework may be done in groups of up to two students.

What to turn in

C. Turn in all related source code (.java files) along with any headers or supplemental libraries needed to compile the code.

D. Use the gradle.build.kts and settings.gradle files that are included in the template repository. Do not change the root project name or version. Doing so will break the autograding script.

E. Do not change the package names or the names of the classes with the main methods in them. You may add classes or methods as necessary.

F. Do not turn in a compiled JAR. The autograder will build the JAR for you automatically.

G. In addition, turn in a README file (*e.g.* README.md, README.txt) with the following:

- Names and TZ of all students in the group
- Total number of hours spent on each part of the assignment
- Date of submission
- Reflections on the assignment's requirements. What did you learn from it? What was the hardest part?

H. Submissions missing any of the above will be penalized 5 points.

How to submit

I. Turn in your submission via the private repositories opened for you on GitHub using GitHub classroom. If you put your code somewhere else or don't use the private repository opened by GitHub Classroom you will not receive a grade!

J. Place the source code for the program in the directory called src.

K. Indicate that your work is complete by performing a single commit with the text "Submitted for grading" on the repository.

- Only write "Submitted for grading" when you are done! I will take the grade from the repository from the first commit with that text.

L. **Do not** send submissions via email. Email submissions will be ignored without consideration of their merits.

Encrypted File Sender

1 Assignment requirements

Your task for this assignment will be to complete and extend the command line encryptor and decryptor tool from the recitations. You will make two tools that use a variety of forms of forms of encryption: An encrypting sender tool and a decrypting receiver tool. The two tools work as follows.

2 Encrypting Sending Tool

The encrypting sender tool receives the following parameters:

dest The IP address of the destination receiver

port The port the destination receiver listens on

infile The file to send (complete or relative path)

iv The IV to use for encrypting the file in hexadecimal format

key The key to use for encrypting the file in hexadecimal format

suite The cipher suite to use for encryption - cipher (AES or DES), mode (CTR, CBC, GCM), and padding technique (None or PKCS5Padding)

taglength The tag length to use for GCM encryption. For other modes, this is optional.

The sending tool encrypts the file using the suite, key, and IV given and sends it to the recipient. When the sending tool is done, it outputs a success line such as: "Sent file file1.pdf" where file1.pdf is the value of the infile parameter.

If there are any parameters missing or any illegal parameters, quit with a usage message.

The autograding script includes many test cases. A few representative examples are shown here.

2.1 Sender Example Tests

2.1.1 Test 1: DES

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5000 -suite=DES/CBC/NoPadding
-infile="tests/TestFile2-oneblock.txt" -key=abcf77646f796f75 -iv=646f7468696e5785
```

Expected output:

```
Sent file tests/TestFile2-oneblock.txt
```

2.1.2 Test 2: DES

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5001 -suite=DES/CBC/NoPadding
-infile="tests/TestFile3-twoblock.txt" -key=abcf77646f796f75 -iv=646f7468696e5785
```

Expected output:

```
Sent file tests/TestFile3-twoblock.txt
```


2.1.3 Test 20: AES

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5019 -suite=AES/CBC/PKCS5Padding
-infile="tests/TestFile9-twobetterthanonequietonthewesternfront.txt"
-key=7A696C6C6F77777377696E7377617273696E746F68656174 -iv=736576656E6A756E696F726D696E7473
```

Expected output:

Sent file tests/TestFile9-twobetterthanonequietonthewesternfront.txt

2.1.4 Test 21: AES

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5020 -suite=AES/CBC/PKCS5Padding
-infile="tests/TestFile10-twobetterthanonequietonthewesternfrontuntiitsoverandtingsbreakup.txt"
-key=7A696C6C6F77777377696E7377617273696E748868656172 -iv=736576656E6A756E696F726D696E7473
```

Expected output:

Sent file tests/TestFile10-twobetterthanonequietonthewesternfrontuntiitsoverandtingsbreakup.txt

2.1.5 Test 37: Sender Error

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5037 -infile="tests/Alaska.pdf"
-key=68756E6472656479656174127761726962E74726F64756374696F6E696E746865
-iv=6974776173746865626573746F666D65 -taglength=128
```

Expected output:

Usage: -infile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding -dest=ip -port=p
iv and key in hexadecimal
suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
dest must be a valid IP address

2.1.6 Test 38: Sender Error

```
java -jar Sender-5785.jar -dest=abcd.0.0.1 -port=5038 -suite=AES/GCM/NoPadding
-infile="tests/Alaska.pdf" -key=68756E6472656479656174127761726962E74726F64756374696F6E696E746865
-iv=6974776173746865626573746F666D65 -taglength=128
```

Expected output:

Error parsing destination address: abcd.0.0.1: Name or service not known
Usage: -infile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding -dest=ip -port=p
iv and key in hexadecimal
suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
dest must be a valid IP address

2.1.7 Test 39: Sender Error

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5039 -suite=ACM/GCM/NoPadding
-infile="tests/Alaska.pdf" -key=68756E6472656479656174127761726962E74726F64756374696F6E696E746865
-iv=6974776173746865626573746F666D65 -taglength=128
```

Expected output:

Invalid algorithm chosen: acm/gcm/nopadding
Usage: -infile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding -dest=ip -port=p
iv and key in hexadecimal

suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
dest must be a valid IP address

2.1.8 Test 40: Sender Error

```
java -jar Sender-5785.jar -dest=127.0.0.1 -port=5040 -suite=AES/GCM/NoPadding
-infile="tests/Alaska.pdf" -key=68756E64726564796561741276 -iv=6974776173746865626573746F666D65
-taglength=128
```

Expected output:

Error: Invalid key when encrypting: Invalid AES key length: 13 bytes
Error sending file: Connection refused

3 Decrypting Receiving Tool

The decrypting receiving tool receives the following parameters:

ip The IP address to receive from

port The port the destination receiver listens on

tempfile The path to write the received encrypted bytes (the bytes received from on the network)

outfile The path to write the decrypted contents (path)

iv The IV to use for decrypting the file in hexadecimal format

key The key to use for decrypting the file in hexadecimal format

suite The cipher suite to use for encryption - cipher (AES or DES), mode (CTR, CBC, GCM), and padding technique (None or PKCS5Padding)

taglength The tag length to use for GCM encryption. For other modes, this is optional.

The receiving tool listens on the IP address and port given and writes the bytes it receives (encrypted) to the tempfile file. When the recipient tool finishes receiving the encrypted bytes it outputs a success line such as: "Received encrypted file: fileenc.pdf" where fileenc.pdf is the value of the tempfile parameter.

Then the recipient decrypts the file using the suite, key, and IV given and writes the decrypted data to the outfile. When the decryption is done, it outputs a success line such as: "Finished decrypting file: file1.pdf" where file1.pdf the vale of the outfile parameter.

If there are any parameters missing or any illegal parameters, quit with a usage message.

The autograding script includes many test cases. A few representative examples are shown here. The primary purpose of the assignment is to learn how to use encryption and decryption correctly, so the receiver's output files (tempfile and outfile) will be checked for correct contents using the sha256sum tool. You can read about the tool by typing "man sha256sum" or at: `https://linux.die.net/man/1/sha256sum`.

3.1 Receiver Example Tests

3.1.1 Test 1: DES

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5000 -suite=DES/CBC/NoPadding
-tempfile="encryptedfiles/Test1-TestFile2-oneblock-DES-CBC-NoPadding-encrypted.txt"
-outfile="decryptedfiles/Test1-TestFile2-oneblock-DES-CBC-NoPadding-decrypted.txt"
-key=abcf77646f796f75 -iv=646f7468696e5785
```

Expected output:

Received encrypted file: encryptedfiles/Test1-TestFile2-oneblock-DES-CBC-NoPadding-encrypted.txt
Finished decrypting file: decryptedfiles/Test1-TestFile2-oneblock-DES-CBC-NoPadding-decrypted.txt

3.1.2 Test 2: DES

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5001 -suite=DES/CBC/NoPadding
-tempfile="encryptedfiles/Test2-TestFile3-twoblock-DES-CBC-NoPadding-encrypted.txt"
-outfile="decryptedfiles/Test2-TestFile3-twoblock-DES-CBC-NoPadding-decrypted.txt"
-key=abcf77646f796f75 -iv=646f7468696e5785
```

Expected output:

Received encrypted file: encryptedfiles/Test2-TestFile3-twoblock-DES-CBC-NoPadding-encrypted.txt
Finished decrypting file: decryptedfiles/Test2-TestFile3-twoblock-DES-CBC-NoPadding-decrypted.txt

3.1.3 Test 20: AES

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5019 -suite=AES/CBC/PKCS5Padding
-tempfile="encryptedfiles/Test20-TestFile9-twobetterthanonequietonthewesternfront
-AES-CBC-PKCS5Padding-encrypted.txt" -outfile="decryptedfiles/Test20-TestFile9
-twobetterthanonequietonthewesternfront-AES-CBC-PKCS5Padding-decrypted.txt"
-key=7A696C6C6F77777377696E7377617273696E746F68656174 -iv=736576656E6A756E696F726D696E7473
```

Expected output:

Received encrypted file: encryptedfiles/Test20-TestFile9-twobetterthanonequietonthewesternfront-
AES-CBC-PKCS5Padding-encrypted.txt
Finished decrypting file: decryptedfiles/Test20-TestFile9-twobetterthanonequietonthewesternfront-
AES-CBC-PKCS5Padding-decrypted.txt

3.1.4 Test 21: AES

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5020 -suite=AES/CBC/PKCS5Padding
-tempfile="encryptedfiles/Test21-TestFile10-twobetterthanonequietonthewesternfront
untiitsoverandtingsbreakup-AES-CBC-PKCS5Padding-encrypted.txt"
-outfile="decryptedfiles/Test21-TestFile10-twobetterthanonequietonthewesternfrontunt
iitsoverandtingsbreakup-AES-CBC-PKCS5Padding-decrypted.txt"
-key=7A696C6C6F77777377696E7377617273696E748868656172 -iv=736576656E6A756E696F726D696E7473
```

Expected output:

Received encrypted file: encryptedfiles/Test21-TestFile10-twobetterthanonequietonthewesternfront
untiitsoverandtingsbreakup-AES-CBC-PKCS5Padding-encrypted.txt
Finished decrypting file: decryptedfiles/Test21-TestFile10-twobetterthanonequietonthewesternfront
untiitsoverandtingsbreakup-AES-CBC-PKCS5Padding-decrypted.txt

3.1.5 Test 41: Receiver Error

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -suite=AES/GCM/NoPadding
-tempfile="encryptedfiles/Test36-Alaska-AES-GCM-NoPadding-encrypted.pdf"
-outfile="decryptedfiles/Test36-Alaska-AES-GCM-NoPadding-decrypted.pdf"
-key=68756E6472656479656174127761726 96E74726F64756374696F6E696E746865
-iv=697477617374686562657374 6F666D65 -taglength=128
```

Expected output:

Illegal port: Cannot parse null string
Usage: -ip=ip -port=p -tempfile=path/to/file -outfile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding
iv and key in hexadecimal
suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
ip must be a valid listening ip
tempfile is where to write received encrypted contents
outfile is where to write decrypted contents

3.1.6 Test 42: Receiver Error

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5042 -suite=ACM/GCM/NoPadding
-tempfile="encryptedfiles/Test36-Alaska-AES-GCM-NoPadding-encrypted.pdf"
-outfile="decryptedfiles/Test36-Alaska-AES-GCM-NoPadding-decrypted.pdf"
-key=68756E6472656479656174127761726 96E74726F64756374696F6E696E746865
-iv=6974776173746865626573746F666D65 -taglength=128
```

Expected output:

Invalid algorithm chosen: acm/gcm/nopadding
Usage: -ip=ip -port=p -tempfile=path/to/file -outfile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding
iv and key in hexadecimal
suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
ip must be a valid listening ip
tempfile is where to write received encrypted contents
outfile is where to write decrypted contents

3.1.7 Test 43: Receiver Error

```
java -jar Receiver-5785.jar -ip=127.0.0.1 -port=5044 -suite=AES/GCM/NoPadding
-tempfile="encryptedfiles/Test36-Alaska-AES-GCM-NoPadding-encrypted.pdf"
-outfile="decryptedfiles/Test36-Alaska-AES-GCM-NoPadding-decrypted.pdf"
-key=68756E6472656479656174127761726 96E74726F64756374696F6E696E746865 -iv=69747876
-taglength=128
```

Expected output:

Error: Invalid IV when decrypting: Invalid AES IV length: 4 bytes
Usage: -ip=ip -port=p -tempfile=path/to/file -outfile=path/to/file -iv=iv -key=key -suite=cipher/mode/padding
iv and key in hexadecimal
suite must be a valid cipher suite (e.g. AES/CBC/NoPadding)
ip must be a valid listening ip
tempfile is where to write received encrypted contents
outfile is where to write decrypted contents

4 Code documentation

Add Javadoc documentation to every method and class. The documentation for methods must include:

- A 1-2 sentence summary of the method's purpose
- @param entries for all parameters, including what they are used for
- @return entry with a 1-2 sentence description of the return value
- @throws entries for any exceptions thrown.

The documentation for classes must includes:

- A 2-3 sentence description of the class' purpose
- @author the code's author
- @version a version for the class. Update on every commit to the repository.

5 Submission notes

Use the assignment starter repository for your code. The repository includes test files in the tests directory and expected output files in the outputs directory. Don't change the input files. You can use the target output files to help test your code. The repository contains autograding tests that will cover most of the grading for the assignment.

5.1 Line Endings

Some tests encrypt and decrypt text files. If you are writing your code in Windows, be sure that your test text files either have no new line characters or have the Unix (LF) line ending format. Windows uses a different line ending format (CRLF) which will give you different results. If you receive errors in testing on Windows, check that all text files have only LF line endings.

Tools such as Notepad++ can help you easily change the line ending format with a few clicks. There are many tutorials on how to do that. One is at: `https://stackoverflow.com/questions/8195839/choose-newline-character-in-notepad`.

6 Grading notes

- Input and Output tests:
  - Success message outputs: 6 points
  - Encrypted and decrypted contents: 74 points
- Javadoc documentation: 10 points
- GitHub usage, Readme, and Reflection: 10 points

