CS306 Practical Lab 2

Symmetric Key Crypto with the Java Standard Library

Welcome to today's lab, students! This lab will be run quite differently from other labs thus far in the semester. You will be using the code provided to perform independent research about different symmetric key cryptography functions.

On canvas, you will find a file called `SymmetricKeyTest.java`. This class contains all of the code necessary to complete this lab. Be sure to read through the code and make sure you understand everything that it does. Feel free to ask questions if needed.

Part 1: Understanding the Program

The supplied program takes 5 arguments:

1. **Algorithm:** this is the encryption algorithm that the program will use. It can take on the following values: **DES** (56 bit), **TripleDES** (112 bit), **RC2** (8-1024 bits, in steps of 8 bits; default 64 bits), **AES** (128, 192, or 256 bit), or **Blowfish** (any multiple of 8 bits between 32 and 448 bit).
2. **Key Size:** The length of key that will be produced by the program. Eah algorithm above has certain key sizes that it accepts. Use one of these values. The program cannot accept key sizes above 128 bits.
3. **Mode of Operation:** As discussed in last week's lab, an encryption algorithm's mode of operation is the method by which the algorithm is applied to consecutive blocks of a message. It can take on the following values: **ECB, CBC, OFB, and PCBC.**
4. **Iterations:** The number of times to run the encryption and decryption algorithms. We recommend a value above 100000 to get a good amount of data.
5. **Message:** The message to encrypt.

It will output the times spent on encryption and decryption on average, as well as a proof that it was correctly encrypting and decrypting messages.

Part 2: Gathering Data

1. Run the program using the same message, number of operations, and mode of operation each time. Use the minimum key size for each algorithm. For each algorithm, take note of the encryption and decryption times.
2. Using the AES algorithm, experiment with each different mode of operation. Take note of the encryption and decryption times.
3. Using the RC2 algorithm on ECB mode, experiment with different key sizes. Try at least 3 different key lengths. Take note of the encryption and decryption times.

4. Using the AES algorithm, experiment with the message length. Try at least 3 different message lengths. Take note of the encryption and decryption times.

Part 3: Questions

1. What happens if you try to use a key size greater than 128 bits in any of the algorithms? Do some research as to the reason for the apparent inconsistency between the algorithm definitions and the Java Standard Library implementations.
2. Why does the RC2 implementation require a 40-bit key minimum, despite the standard specifying an 8-bit minimum? Intuitively, what is the problem with an 8-bit key (hint: how many bits make up an ASCII character)?
3. Which algorithm was the fastest? The slowest? What is a scenario in which a slower algorithm may be better?
4. Which mode of operation was the fastest? The slowest? What is the benefit to using one of the chaining modes as opposed to ECB? How does PCBC differ from CBC?
5. One of these algorithms is the accepted standard in the US. Which one? Why?
6. How did key length impact processing time? What about security?
7. Did changing the message length impact the processing time? Let's assume instead that the application was multithreaded and using the ECB mode of operation. What impact would this optimization have on longer messages?
8. We have hard-coded the PKCS5Padding algorithm into the program. What does this algorithm accomplish?