King Fahd University of Petroleum and Minerals
College of Computer Sciences and Engineering
Department of Computer Engineering

COE 451 – Computer and Network Security (T201)

Mini-Programming Project-Phase II

(due date: Sunday 08/11/2020 by 11:59 PM on Blackboard)

LATE SUBMISSIONS WILL NOT BE ACCEPTED

Description:
Modify the *online chatting* application that you had for phase 1 so that your code encrypts/decrypts the data exchanged between the client and the server. **Use AES symmetric-key crypto system with 256-bit key and with CBC mode in your code**. Use an online SHA256 to hash your student ID to generate a unique 256 bits hashed value to be used as the AES 256-bit key. For this phase, you can hard code the 256-bit key in your client and server applications. As for the Initialization Vector (IV) to be used by the CBC mode, make sure to generate **a new IV for each new chat session**. The side that **initiates** the chat session is responsible for generating the IV and should send the IV to the **other** side before data is exchanged between the two sides. The IV should be 128 bits long to match the AES plaintext block size, and it should be generated using a **cryptographically secure** random number generator (**CSRNG**) function/method (research which **CSRNG** function/method is the most suitable to use for your code through the Internet). The **receiving** side will always expect the first 16 bytes (or 128 bits) that are received to contain the IV. To encrypt data to be sent to the other side, you may need to pad the data if the last generated plaintext block of the data is less than 128 bits long (research the proper way of how padding should be done through the Internet).

Phase 2 must be your own genuine code. You may use existing libraries/methods/functions but only for AES and CSRNG; the rest of the code needed for phase 2 must be developed by you.

Make sure to test your modified application preferably using 2 virtual machines. Verify your code by comparing the ciphertext that your application generates against an equivalent ciphertext generated by some of the existing online AES-256-CBC tools.

Note: Some of the online AES-256-CBC tools will pad the plaintext even if it was a multiple of 128 bits while others don't. So, if your padding implementation is not similar to that of the online tool, then there will be a mismatch between only the last part of your ciphertext and the ciphertext generated by the online tool.

Deliverables: Submit the following items to Blackboard before the due date:

1. A **well commented source code** of your implementation. Highlight the code that you specifically added to this phase.
2. A **step-by-step readme file** that explains what is needed to execute your implementation and how to execute and test your implementation.
3. A **short soft copy report** that contains the following:
   i. Screenshot of the unique 256-bit key used by your code.
   ii. Screenshot of the plaintext data exchanged between the two sides. Show this information for at least 2 different chat sessions.
   iii. Screenshot of the corresponding ciphertext along with the IV used for the exchanged data. Show this information for at least 2 different chat sessions.
   iv. Screenshot of the corresponding verification with an online tool. **Make sure to specify the URL of the online tool that you use for verification.**

Grading Scheme:

Program readability and comments	[15 points]
Step-by-step readme file	[15 points]
Screenshot of the unique 256-bit key used by your code	[10 points]
Screenshot of the plaintext data exchanged between the two sides (for at least 2 different chat sessions)	[10 points]
Screenshot of the corresponding ciphertext along with the IV used for the exchanged data (for at least 2 different chat sessions)	[25 points]
Screenshot of the corresponding verification with an online tool	[25 points]
Total	[100 points]