**DFSC 3316 Lab 1 Burris Spring 2023**

**Lab 1: Due Wednesday March 8 prior to the start of class. Expect to receive additional assignments during this time.** You may use university or personnel equipment. You may not use code from any source other than the notes without my expressed written permission. *Use of outside resources including family, friends, tutors, or paid help will result in an “F” for the class. You are however encouraged to discuss strategy, how you solved problems, and provide reasonable comfort/aid to other members of the class*. Submit printed copies of all code, data files, and other pertinent materials. Your lab should be submitted in a 8-1/2 by 11 mailing envelop. You may use the same envelop for all labs during the semester. **I expect you to demonstrate to me your system works properly prior to the due date!**

To master the field of Digital Forensics, you must be knowledgeable in many aspects of digital communications. I recommend the use of Java for this lab as it will minimize your learning curve. You may however use other languages such as “C” by asking permission prior to starting the lab.

Write this lab in a highly modular fashion. Expect to potentially implement essentially the same lab at a future date using servlets, SSL, and/or RMI. You should consider reuse of your code as a major goal! You need not concern yourself with the man-in-the-middle attack on this lab

An interesting application requiring authentication is logging in to a computer network, host computer, or automated money machine (ATM). Generally, a user is required to supply a user name and corresponding password. The host creates and stores a table of usernames and corresponding passwords. If an unscrupulous user or hacker obtains access to the server and copies the table off the server, ATM, or backup media, they now have substantial leverage. In 1969, Roger Needham and Mike Guy as reported in Chapter 1 of “*Authentication: From Passwords to Public Keys*,” by Richard E. Smith (9/12/2001) ISBN 0-201-61599-1, Addison Wesley Longmire Inc) noted a serverreally does not need to know the passwords; it just needs to be able to differentiate valid passwords from invalid passwords. Their technique is known as **Authentication Using One-Way Functions** and is widely utilized especially by operating systems. The technique proceeds as follows:

**Storing the one-way function:**

**As legitimate users and passwords are entered into the system, a one-way hash is computed for the password. The user name and hash are stored for future reference. The password is discarded.**

**Access to the system:**

1. **To login, the user enters their user name and password.**
2. **The host computes a one-way hash of the password then discards the actual password.**
3. **The host now compares the one-way hash to the previously stored value of the hash associated with the account name. If they are the same the user is authenticated.**

**“C/B” Option (best grade is 89):**

Please implement a password system for a manufacturer storing at least 10 user names and password hashes in an array or file (preferred) as described by Needham and Guy. You may create the contents of the array using any industry standard hash.

Once users are authenticated, they wish to be able to exchange an arbitrary number of communications with the manufacturer using symmetric encryption. The following protocol has been agreed upon to determine a synchronous encryption algorithm and secret key. You should be able to dynamically add new user and combinations.

Protocol:

1. **Using a socket**, customers contact the server (manufacturer) and login in for validation using their account name and password.
2. Upon validation, the manufacturer and customer determine a secret session key via a two part process. First they determine an initial secret key based on the work of Diffie-Helleman. It has been decided to use the SKIP protocol with a 1024 bit key. You may use a different protocol than Diffie-Helleman with permission from the instructor. Once the initial key has been determined, you may utilize hashing or another technique of your choice to reduce it to a size appropriate (number of bytes) to initialize the symmetric key encryption algorithm. The rest of the communications are completed using symmetric encryption.
3. The client should determine the symmetric algorithm to be used and communicate their choice to the server. The server should support at least three symmetric encryption algorithms. You are free to take advantage of code from examples used in class as long as you document the source (do not plagiarize). You may not use code from other sources!

Implement the client and server. Both client and server should print sufficient material to show they have:

1. Properly completed the login protocol.
2. Generated the same initial key using SKIP (or alternate technique).
3. The client must encrypt a file and transmit it to the server. The server must then decrypt the file. Yo**u must convenience me this communication has been accomplished during a live demonstration** or sufficient information to convenience me you accomplished the task.
4. The server must encrypt a file (**with different contents**) and transmit it to the client. The client must then decrypt the file. You may be asked to demponstrate your software in a live demonstration.

**“A” Option (best grade is 100):**

**Do either option 1, 2 or 3.**

Option 1:

Modify the “C/B” option as follows. The server must send an email to an address supplied by the client acknowledging the day and time the conversation was initially obtained from the system clock.

Option 2:

Implement the “C/B” option but store the table of user names and password hashes in a database as opposed to an array. You will be expected to access the database via SQL statements in your code. You must convenience me your design and implementation are three-tier.

Option 3:

Implement the “C/B” option but the server must create a separate instance of itself to service each client (multi thread or process). You must convenience me during your demonstration that you can support multiple clients simultaneously.

Hint for “A” Option 3: If you place a read statement in the server code, it will have to stop and wait on other processes till the desired data is available. The input need not be relevant other than to force the wait condition.

**Preference:**

I prefer a live demonstration of your code. I am willing to accept a printout of your session validating you code accomplishes the desire tasks. Use highlighters or other techniques to document where each requirement is met in the printout.

**You must write all code other than the hash and ciphers. You may use Microsoft Windows, Linux, Apple or other operating system. If you write your code in Java, you should be able to execute the server and client on multiple different operating systems sharing the network.**

Please consider using hash and/or ciphers other than those taught in class like SHA256 and Ghost.