Assignment 2

Goal

The goal is the second assignment is to familiarize students with the use of cryptographic libraries. This is an assignment for **individuals** - no group work or collaboration allowed. Violations of the FIU honor code will be reported to the University immediately, and students will fail the course.

Description

Due: October 7th, 2025, via Canvas *by* 10:00AM EDT.
Points: 80

In this homework you will create file encryption/decryption/transmission suite akin to scp (this is a tool in Linux using ssh protocol for fill transfer) using *OpenSSL* libraries provided by the Linux operating system. Details of this assignment are given below.

- The programs are to be written in 'C/C++' and use the *openssl* library. The *make* or *cmake* utility must be used to create the program.
- The file encryption programs *gsend* and *grec* should take the following inputs:

  ```
  gsend <input file> [-d < IP-addr:port >][-l]
  grec <filename>  [-d < port >][-l]
  ```

  where gsend takes an input file and transmits it to the IP address/port specified on the command-line (-d option), or dumps the encrypted contents of the input file to an output file of the same name, but with the added extension ``*.FIU"* e.g., if the input file is *hello.txt*, the output file should be *hello.txt.FIU*. Note that the grec should run as a network daemon (-d), awaiting incoming network connections on the command-line specified network port. When a connection comes in, it writes the file data to "filename" and exits. grec can also be run in local mode (-l) in which it bypasses the network functionality and simply decrypts a file specified as input. It is assumed that the input file (for decryption) ends ".FIU", so the output will be the original filename without this additional extension. (This is simply the inverse of gsend).

- On each invocation, *gsend* and *grec* should prompt the user for a password. This password will be used to securely generate an encryption using PBKDF2

(Password Based Key Derivation Function 2). When running PBKDF2, use SHA-512 with 4096 iterations and the string "KCl" as the salt. *openssl* implements PBKDF2, so use their implementation. After generation, for grading the key should be printed as a hexadecimal number as in the example execution below.

- Encryption must be done using AES256 in Cipher Block Chaining (CBC) Mode. Be sure to initialize the IV of each invocation to a cryptographically generated pseudorandom number.
- In addition to encryption, gsend and grec should use an HMAC for authentication. On encryption, the HMAC should be appended to the output, and on decryption it should be removed before writing the output. grec should give an error and exit (with exit code 62) if the input HMAC is not correct. Use SHA-512 as the hash function, and HMAC the *encrypted* data (Encrypt-then-MAC). OpenSSL provides SHA-512 and provides a flag to use it in HMAC mode. Use the same key as encryption (for simplicity).
- Both *gsend* and *grec* should display an error and abort the operation if the output file already exists. main() should return 33 when this happens.
- To turn in: you will create a tar file named < *lastname* >*-assign2.tgz* which contains a single directory < *lastname* >*-assign2* which contains the Makefile and all of your source code for the program. All source code and the makefile should be extensively commented. I should be able to run the following commands (after retrieving the file from Canvas):
- `tar xvfz < lastname >-assign2.tgz`
- `cd < lastname >-assign2`
- `make`
- `cp gsend testfile.check`
- `./gsend testfile.check -l`
- `rm testfile.check`
- `./grec testfile.check.FIU -l`
- `diff testfile.check gsend`

Warning: If I cannot do these minimal operations successfully, you will not receive a passing grade. This is not all I will test---I have another battery of tests I will do when grading including carefully reviewing your code. While developing on any *nix system is fine, your code *must* run correctly on Linux. I recommend testing on a Linux machine or VM before submission.

Note: Like all assignments in this class you are prohibited from copying any content from the Internet or sharing ideas, code, configuration, text or anything else or getting help from anyone in or outside of the class. Consulting online sources is acceptable, but under no circumstances should anything be copied. Failure to abide by this requirement will result in immediate reporting to the University.

An example session using these tools *might* look like (this is not actual output):

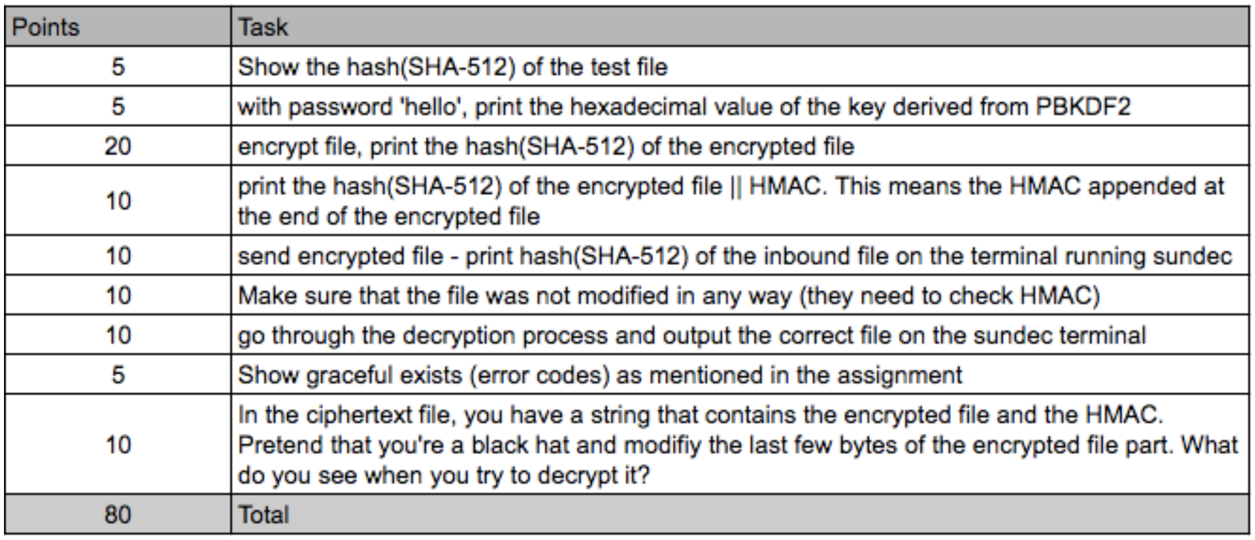
```
alice% ./gsend testfile -d 192.168.1.48:8888
Password: example
Key: FA 3A 2F D3 7E CB 27 F2 B3 42 35 06 CF B1 ED 0E
Successfully encrypted testfile to testfile.FIU (6512 bytes written).
Transmitting to 192.168.1.42:8888
 Successfully received

-----

bob% ./grec testfile -d 8888
Waiting for connections.
Inbound file.
Password: example
Key: FA 3A 2F D3 7E CB 27 F2 B3 42 35 06 CF B1 ED 0E
(this is not the expected output)
Successfully received and decrypted testfile (6400 bytes written).
nittany%
```

Submission Instructions

Students must turn in their tar file to Canvas by 10:00AM on October 7th, 2025. Students must also turn in a *written report* that demonstrates all of the tasks in the rubric below. Students should run their tests using the assignment description file as input.



Points	Task
5	Show the hash(SHA-512) of the test file
5	with password 'hello', print the hexadecimal value of the key derived from PBKDF2
20	encrypt file, print the hash(SHA-512) of the encrypted file
10	print the hash(SHA-512) of the encrypted file || HMAC. This means the HMAC appended at the end of the encrypted file
10	send encrypted file - print hash(SHA-512) of the inbound file on the terminal running sundec
10	Make sure that the file was not modified in any way (they need to check HMAC)
10	go through the decryption process and output the correct file on the sundec terminal
5	Show graceful exists (error codes) as mentioned in the assignment
10	In the ciphertext file, you have a string that contains the encrypted file and the HMAC. Pretend that you're a black hat and modifiy the last few bytes of the encrypted file part. What do you see when you try to decrypt it?
80	Total

A note about this assignment

The purpose of this assignment is to introduce familiarity to using cryptographic libraries in a somewhat realistic exercise. Because familiarity is the goal, certain steps have been simplified at the cost of security. One example of weakness is the fact that the PBKDF2 salt is not a random string. Yet another is using the same key for encryption and authentication -- this is not a recommended practice because the compromise of one mechanism means that the other automatically fails. Making these specific changes would in fact make your application robust in practice.