Assignment Discussion

For assignment 1, I implemented an S/Key one-time password system using C++. The basic design is that there are two programs, the client and the server. In the client, the user can initialize a plain text password, and the program will output a one-time password. The client also has the option to output a one-time password if the user has already initialized a plain text password. After the user enters the passwords in plain text, the program stores that password in a file, so when the user needs a new one-time password, the program can grab the plain text password and generate a new password. Also, every time the client program creates a new password, it stores it in two files. One file is where the S/Key system stores all of the generated one-time passwords; this is so that the system can compare future one-time passwords making sure it never outputs the same password twice. The second file stores the most recent generated password. The Second program accepts user input and compares the input to the file that has the most recent one-time password. If the two strings match, the program outputs "SUCCESSFULLY AUTHENTICATED," and if it fails, the program outputs "FAILED TO AUTHENTICATE."

The reason I chose this model for my assignment is because of its simplistic approach. It makes it easier to anticipate possible communication problems between the two programs. Since the two programs pull information from external files, certain special cases can occur. For example, if a user tries to generate a new one-time password, but the program cannot find one of the three files mentioned earlier than the program outputs an error message. The message states that the user needs to initialize a new password because one or more files are missing. By initializing a new password, the program re-creates all three storage files. If a user tries to access the server and no file has the most recent password, then the program outputs "FAILED TO AUTHENTICATE."

The crypto technique used was a C++ library hash function. The library hash function I used generates a random 20-bit number. The reason I chose this crypto technique is that it is fast and straightforward for a person to type in by hand. The tradeoff is that it is less secure than say a 256-bit secure hash function. Obviously, the longer the one-time password is, the better the security. When designing my S/Key, I chose to lean more towards accessibility rather than security. In conclusion, my assignment performs as excepted and meets all the requirements. Below you can view the test cases I used to test my S/Key sysetm.

Test Cases:

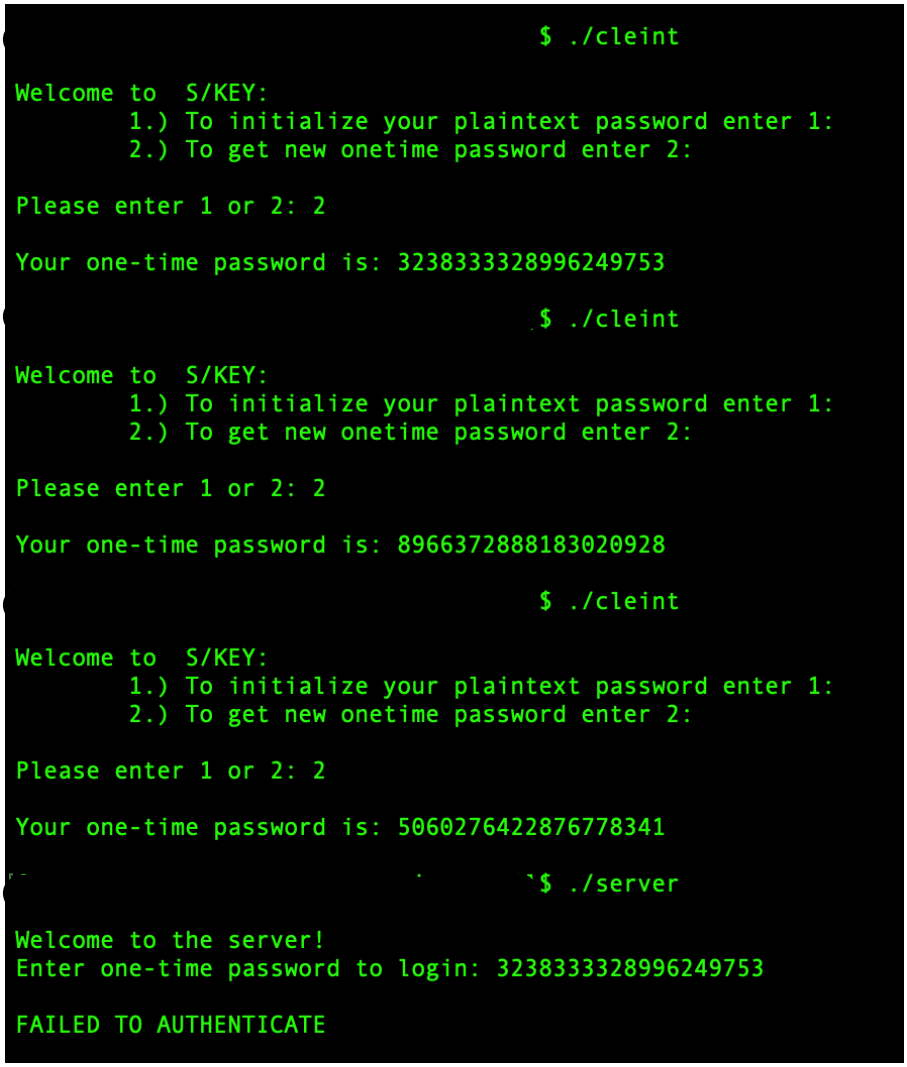
1.

```
$ ./cleint

Welcome to  S/KEY:
        1.) To initialize your plaintext password enter 1:
        2.) To get new onetime password enter 2:

Please enter 1 or 2: 1
Type your plaintext password: testTest

Your one-time password is: 3675770805988527390

$ ./server

Welcome to the server!
Enter one-time password to login: 3675770805988527390

SUCCESSFULLY AUTHENTICATED
```

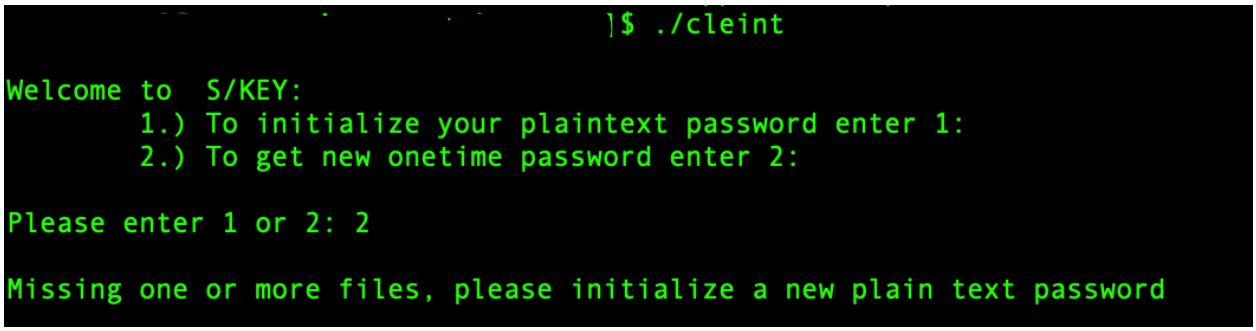
2.

```
$ ./cleint

Welcome to  S/KEY:
        1.) To initialize your plaintext password enter 1:
        2.) To get new onetime password enter 2:

Please enter 1 or 2: 2

Your one-time password is: 15642531337315160639

$ ./server

Welcome to the server!
Enter one-time password to login: 15642531337315160639

SUCCESSFULLY AUTHENTICATED
```

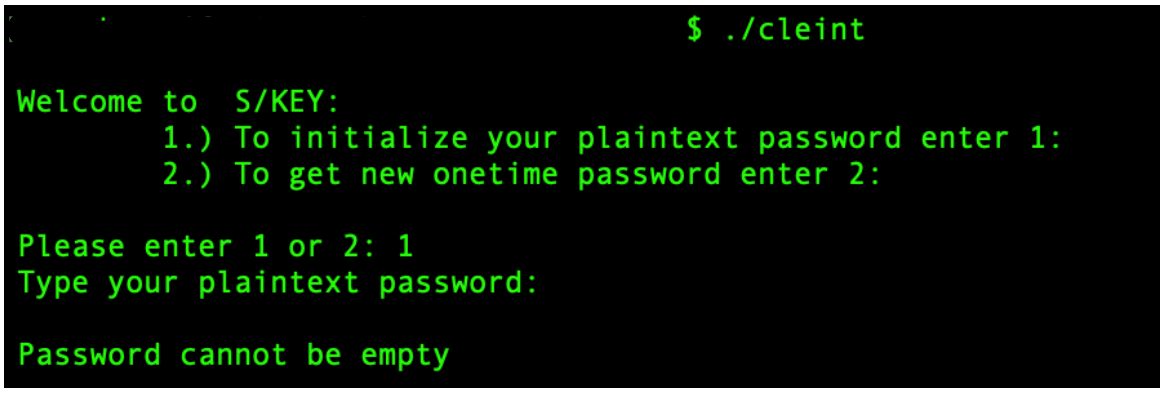
3.


$ ./cleint
Welcome to S/KEY:
1.) To initialize your plaintext password enter 1:
2.) To get new onetime password enter 2:
Please enter 1 or 2: 2
Your one-time password is: 3238333328996249753
$ ./cleint
Welcome to S/KEY:
1.) To initialize your plaintext password enter 1:
2.) To get new onetime password enter 2:
Please enter 1 or 2: 2
Your one-time password is: 8966372888183020928
$ ./cleint
Welcome to S/KEY:
1.) To initialize your plaintext password enter 1:
2.) To get new onetime password enter 2:
Please enter 1 or 2: 2
Your one-time password is: 5060276422876778341
$ ./server
Welcome to the server!
Enter one-time password to login: 3238333328996249753
FAILED TO AUTHENTICATE

4.


$ ./cleint
Welcome to S/KEY:
1.) To initialize your plaintext password enter 1:
2.) To get new onetime password enter 2:
Please enter 1 or 2: 2
Missing one or more files, please initialize a new plain text password

5.


$ ./cleint
Welcome to S/KEY:
1.) To initialize your plaintext password enter 1:
2.) To get new onetime password enter 2:
Please enter 1 or 2: 1
Type your plaintext password:
Password cannot be empty