CSC 4420 Operating Systems

Technical Report

Rihat Rahman

Introduction

At this age of digital technology, there is an increasing demand of devices that offer better services while maintaining the quality. To provide that, there has been a boom in innovation of devices with higher storage capability, better processing time and better reliability. For example, the computers in 1960s had a storage capability of just a few megabytes and were massive in size whereas nowadays a smartphone has more thousands time storage than those and can operate at a very high speed. Even though the supply of devices with better functionalities have increased exponentially, there is still a shortage in the storage capabilities of these devices. The demand of higher storage has surpassed the supply by a large margin which resulted into the invention of cloud storage. Cloud storage is a way to store data online using servers that are provided by the cloud services. There are multiple companies which provide cloud service. Starting from Google, to AWS, Apple iCloud and various other companies provide cloud services.

Although services were able to solve the storage problem of devices, it bought a new problem with itself: security. Storing data online makes in vulnerable to the hackers. It is possible for someone to hack the log in information and log in to manipulate, delete or steal data from the databases. There are healthcare companies, military intelligence, and businesses which store sensitive information on the cloud. Leaks of these confidential information can result in devastating consequences. Encrypting the data is a way to prevent exploitation of leaked information.

Encryption is the process to convert the original representation of the into a form. Decryption is the process to convert the encrypted data back into to the original form. By following this way, only the people who encrypted a file will have the access to decrypt the file. Even if others get able to download the file, they would not to be able to use it as the original information is not displayed in the encrypted file. As mentioned above, there are multiple companies that offer cloud storage services, but for the purpose of this project, I have used Amazon S3 which is a simple storage service offered by Amazon Web Services(AWS). There are also various ways to encrypt a file but I have used OpenSSL rc4 which is discussed on depth later in this technical report.

In this technical report, I will go over my system information, software/packages/tools I used, and the explanation of my code. I will also state the goals of the project, accomplishments, useful lessons, and what I can do to improve my code and the project overall. As this class was more based on practical work, it helped me to teach myself how to learn by my own by making mistakes and doing something over and over again. Finally, I will write a summary talking about what I can take from this class for my future courses and career.

Project Goals

The main goal of this project is to integrate rc4 encryption Amazon’s s3fs cloud system so when a user uploads a file to the cloud, it will be encrypted automatically. Similarly, the file will be decrypted automatically if the user decides to download it into their local folder. The project goals were divided into three parts. The first part was the installation and mounting of s3fs. I made an account in AWS where I created a bucket. From there I downloaded and installed the s3fs package on my machine. After that, I mounted the bucket folder in my local machine which I had created earlier in AWS. The second part of the project was implementation of my own rc4 that will encrypt and decrypt a file as OpenSSL rc4. The function is given a key, input filename, output file, salt/nosalt, and info about whether it is encrypting or decrypting a file. Using different keys would not work with the function. The standalone function should also be compatible with rc4 which means that it should be able to decrypt a file that is encrypted using OpenSSL rc4 and a file it encrypts should be able to get decrypted by OpenSSL rc4, using the same key.

System Information

System Specs (VMware Horizon)

I started my project using VMware Horizon that is offered free to students by Wayne State University. I used it to test OpenSSL rc4 and find out how rc4 works. I also started writing my standalone here as the other virtualization service I was using, Oracle VM Virtualbox was very slow. I could not test my own rc4 functions in VMware Horizon because it did not support all of the characters that were generating from the encryption.

* Operating System: Ubuntu 16.04 LTS
* Memory: 3.9 GB
* Processor: Intel® Xeon(R) CPU E5-2695 v3 @2.30GHz × 2
* OS type: 64-bit
* Disk: 65.4 GB
* Architecture: x86\_64
* CPU(s): 2
* Thread(s) per core: 1
* Core(s) per socket: 1
* Socket(s): 2
* Virtualization type: full

System Specs (Oracle VM Virtualbox)

Virtualbox is a hosted hypervisor for virtualization, developed by Oracle. I had installed linux in my flash drive, but the touchpad of my computer was not supported. Although I had also used VMware Horizon while getting started with the project, but I could not continue to use it as there were problems with backups and data getting deleted every time I log out. For these reasons I chose Virtualbox as my next approach. I installed and mounted s3fs here and did the rest of the work using this.

* Operating System: Ubuntu 18.04.4 LTS
* Memory: 1.9 GB
* Processor: Intel® Core i3-6100U CPU @2.30GHz
* OS type: 64-bit
* Disk: 10.5 GB
* Architecture: x86\_64
* CPU(s): 1
* Thread(s) per core: 1
* Core(s) per socket: 1
* Socket(s): 1
* Virtualization type: full

FUSE

FUSE, also called as Filesystem in Userspace, is an interface which lets users create file systems.

Version: 2.9.7

Amazon s3fs

S3fs allows the mounting of an Amazon S3 bucket by using FUSE which means that, if a file is moved to the s3fs folder, then it will also be uploaded into the S3 bucket online.

Version: 1.86

OpenSSL

OpenSSL library offers secure connections over computer networks. OpenSSL rc4 is used for this project which is a stream cipher and variable length key algorithm. OpenSSL library can be found on:

<https://github.com/openssl/openssl>

Version 1.0.2

Tools and Packages

Oracle VMware Virtualbox

Virtualbox is a virtualization system developed by Oracle that I used go get access to linux operating system. The software can be downloaded from the following link:

https://www.virtualbox.org/

Version: 6.1.6

VMware Horizon

VMware Horizon also helped to use a virtual machine with linux operating system. VMware clients can be downloaded from the following link:

<https://my.vmware.com/en/web/vmware/info/slug/desktop_end_user_computing/vmware_horizon_clients/5_0>

Ubuntu

Ubuntu is a Debian based linux distribution for personal computers. If you use a virtualization software, then you do not have install it in your computer. Otherwise, you will have to install it directly in your computer or flash drive. Latest version of Ubuntu can be downloaded and installed from the following link:

https://ubuntu.com/download/desktop

gedit

gedit is a text editor that was used to write test files to test and edit source files.

Microsoft Visual Studio

Vistual studio is an IDE used to write and compile source files. In this project visual studio was only used to write source file as compilation had required other components. Visual studio can be downloaded from the following link:

<https://code.visualstudio.com/download>

Amazon s3fs

S3fs helps to mount an S3 bucket in linux. The first step to create a local S3 folder is to create an Amazon Web Services account at <https://aws.amazon.com/> and create a bucket. The rest of the downloading, installation, and mounting process can be followed from the following link:

https://youtu.be/F0Rz1xWKtiI

GCC (GNU Compiler Collection)

GCC is used to compile C files. GCC is included in Ubuntu build-essential meta package by default.

Linux Terminal

Linux terminal is used in this project to do tasks like copy, moving, comparing files etc. from the console. It is also required to compile and test standalone rc4 file.

Design

A close up of text on a white background

Description automatically generated

As it can be seen on the diagram above, the EVP\_BytesToKey() is called first which takes EVP\_sha256(), password and other parameters. EVP\_BytesToKey() function will obtain the key from these parameters. After that, RC4\_set\_key

Is called which will set up the key. Followed by that, RC4() is called that will encrypt or decrypt the data based on input.

Integration

Firstly, I installed s3fs in my machine and mounted a bucket. The following commands can be used to mount or unmount a bucket:

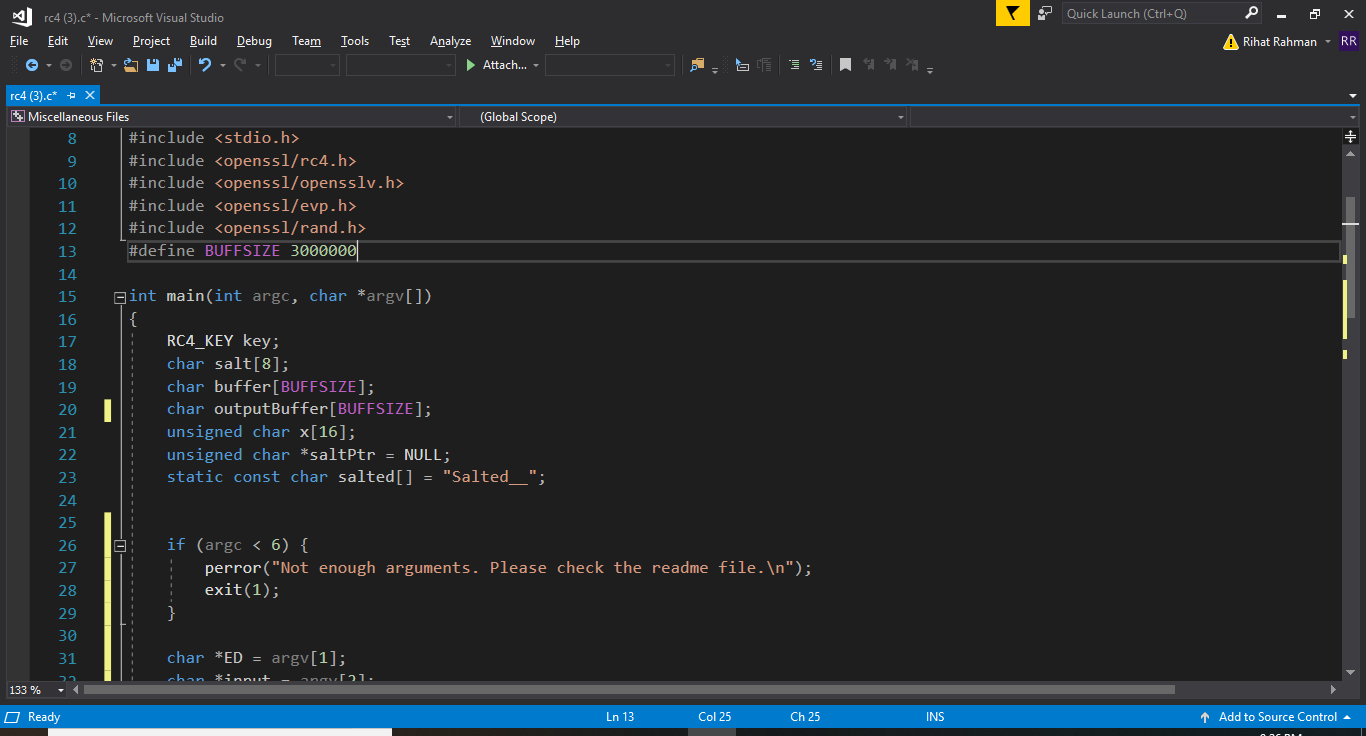
Mount: sudo s3fs -o allow\_other a-test-bucket-124568d /s3/bucket-test

Unmount: umount /s3/ bucket-test/

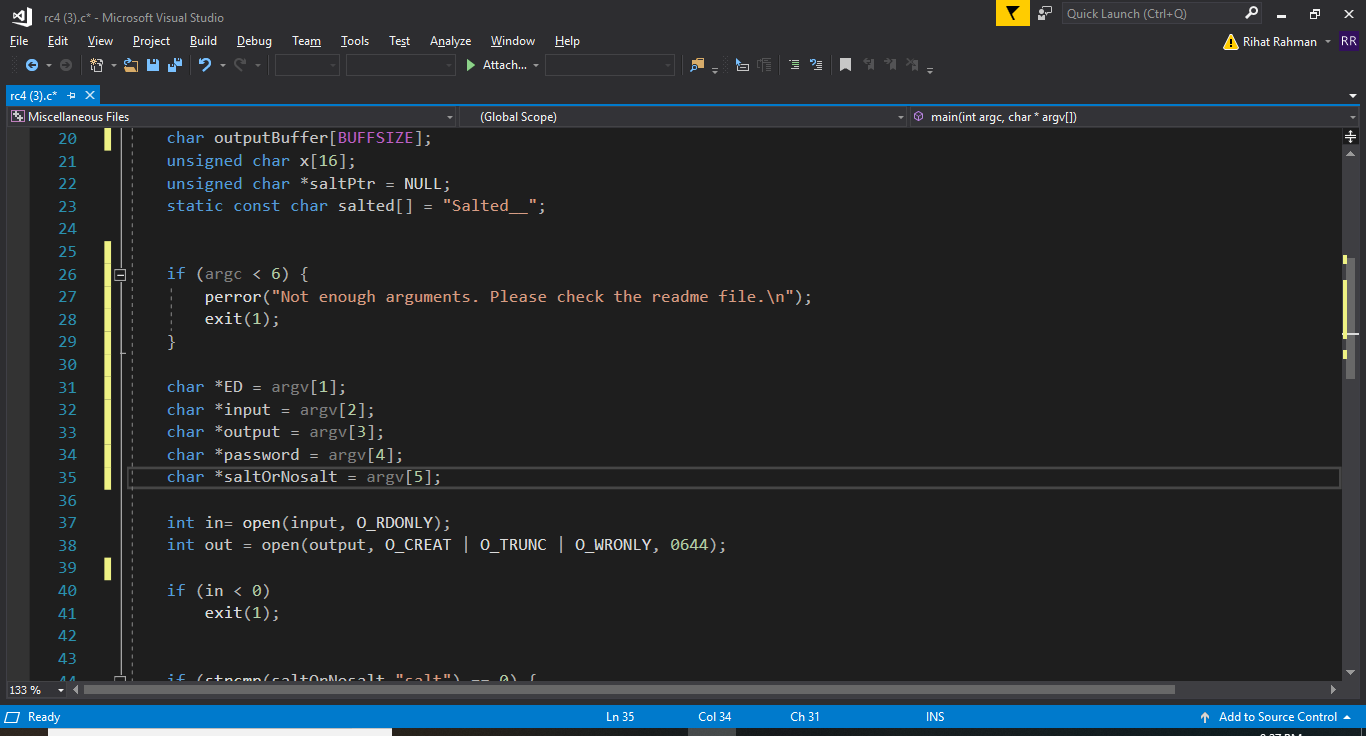
Then I tested by uploading and downloading (moving files in the bucket and out of it). After that, I implemented my own RC4. It worked perfectly by itself, with and without salt. When I started testing the compatibility of my standalone RC4 with salt it worked fine but the problem occurred when I tested the compatibility of my standalone RC4 with salt. It was working fine with some files, but most of the it was not being able to decrypt files encrypted by OpenSSL rc4 and the other way. I spent a lot of time trying to make the salt part compatible with OpenSSL but could not figure it out. I was getting a bad magic number which was probably due to wrong header format as pointed out by Dr. Xu. I tried fixing that but continued getting other errors. After that, I started the integration part by reading s3fs.cpp and fdcache.cpp. Initially, it was difficult to make a connection between the different source files. Initially, I thought that it was the load function in fdcache.cpp that uploads files in the bucket. After that, I made a void rc4 function in fdcache.cpp and called the function from int FdEntity::Load() but did not get any success. After some attempts, I figured out that I will have to call rc4 function from the read and write functions in fdcache.cpp(ssize\_t FdEntity::Read(),ssize\_t FdEntity::Write()). I made a void rc4 function and called it from the write function but it the result did not turn out to be as expected. Because of spending much time on salt compatibility of standalone RC4, I didn’t have enough time to correct my code for s3fs integration.

Implementation Details

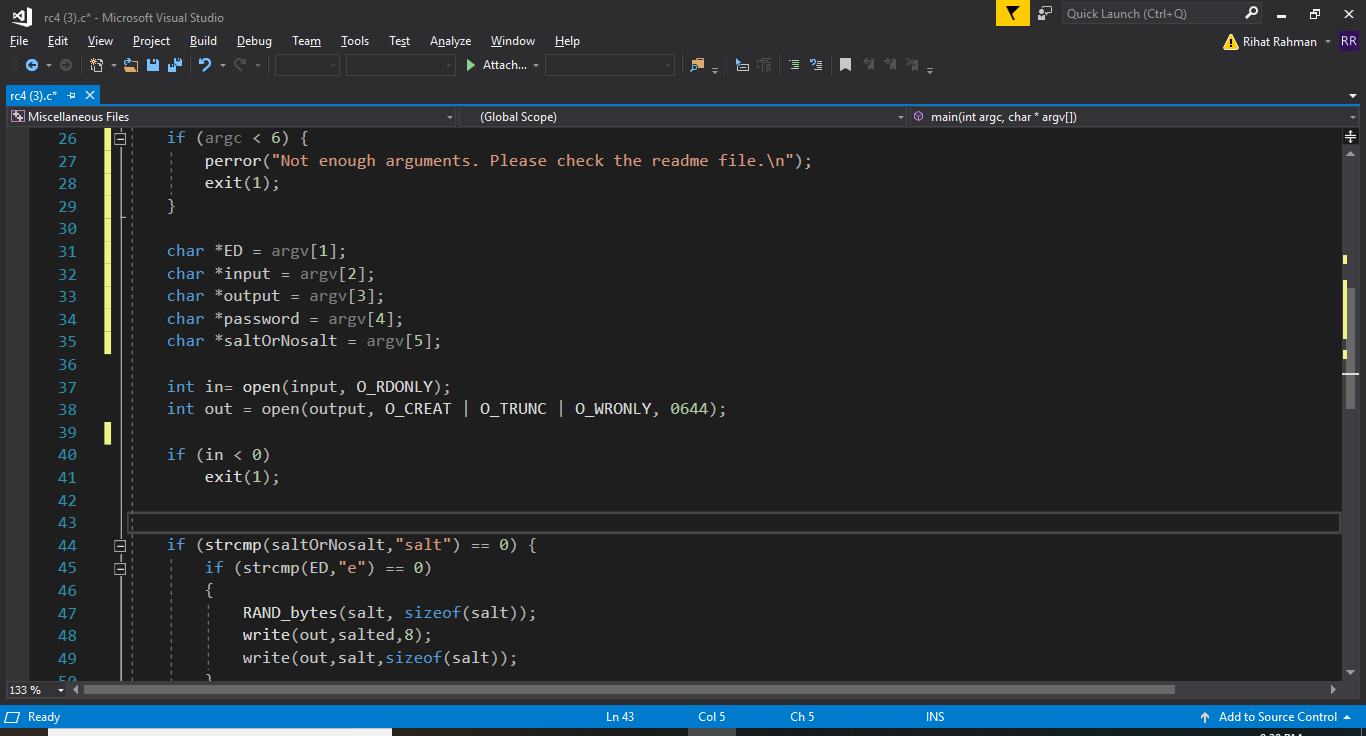
rc4.c



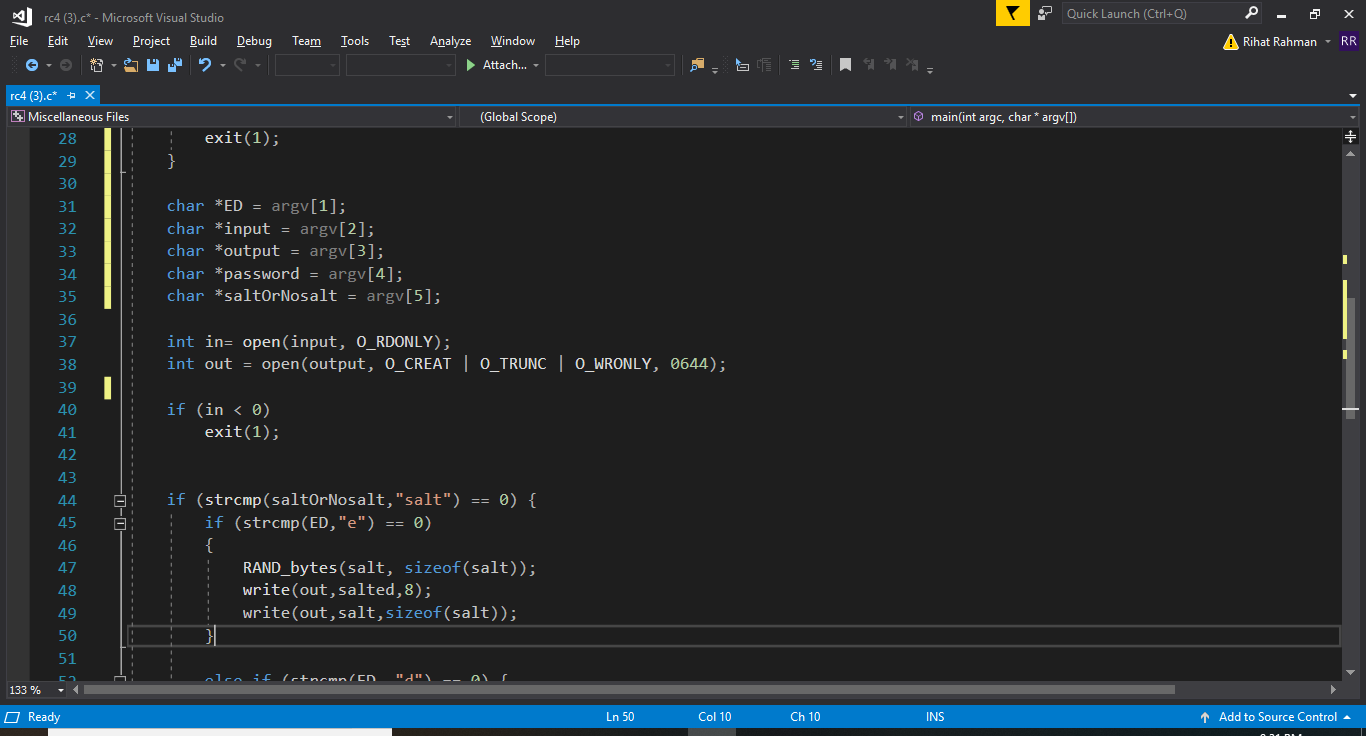
Declare the variables.



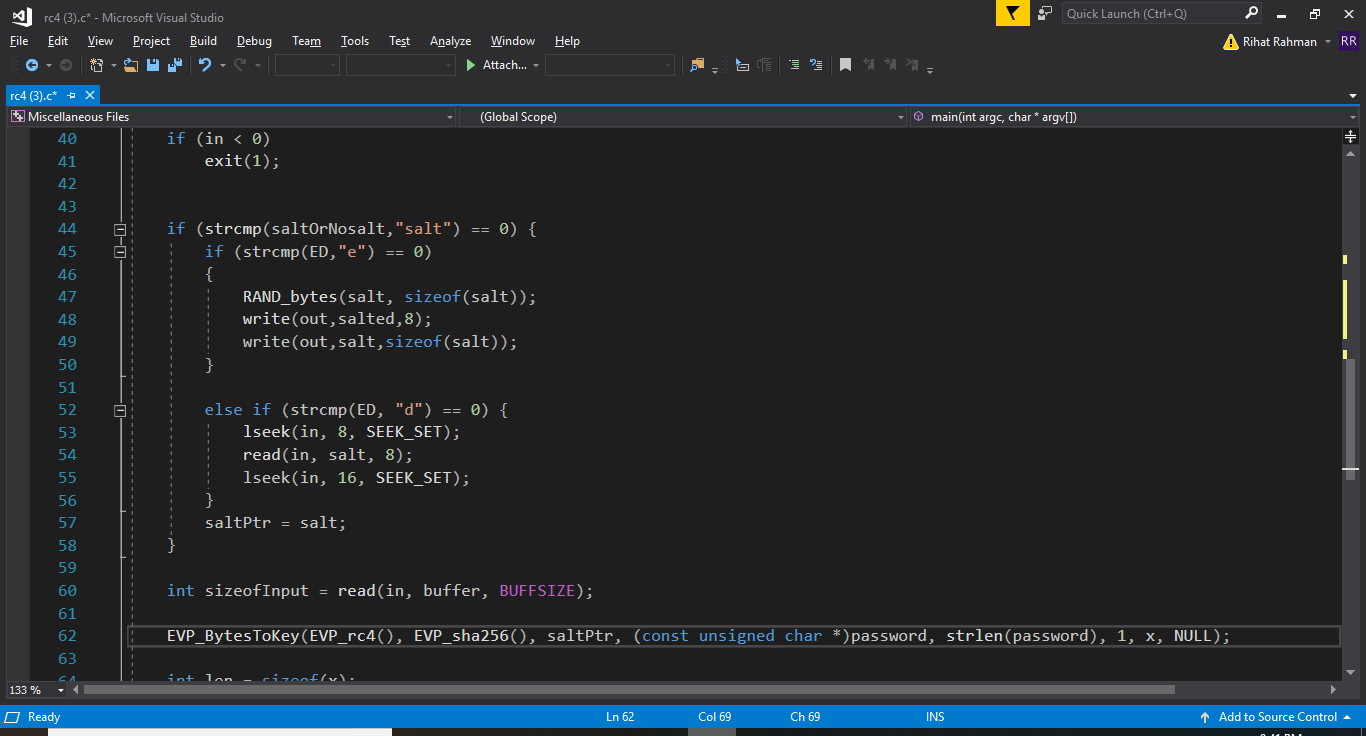
To check if the user provided enough arguments/parameters



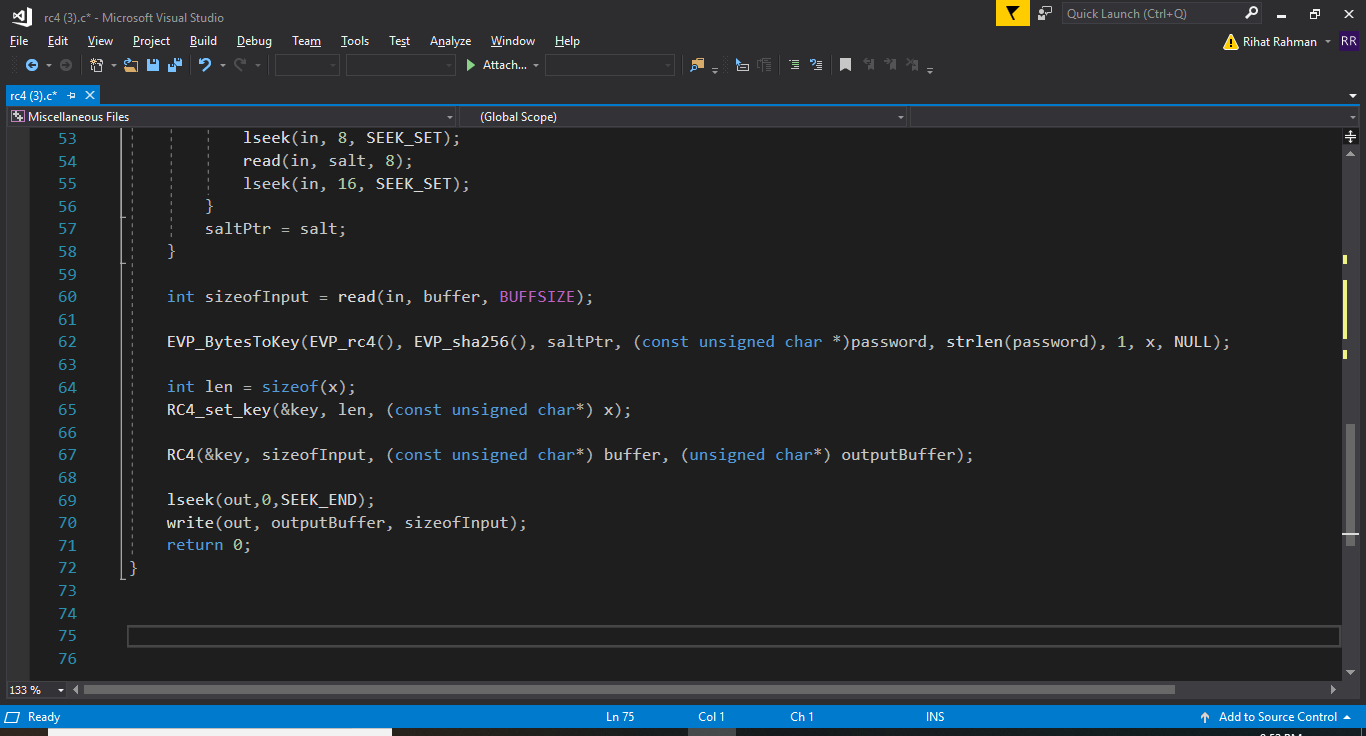
Set variables equal to the parameters



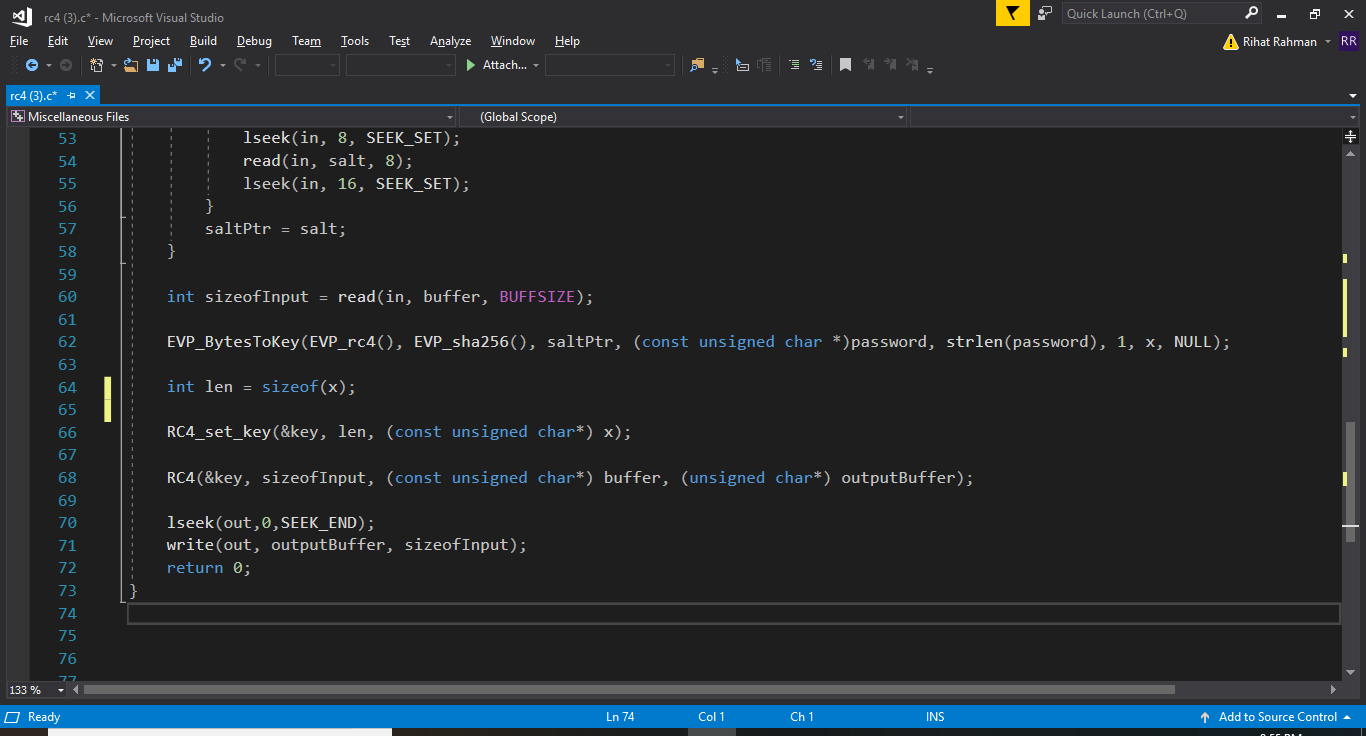
To see if the input file exists



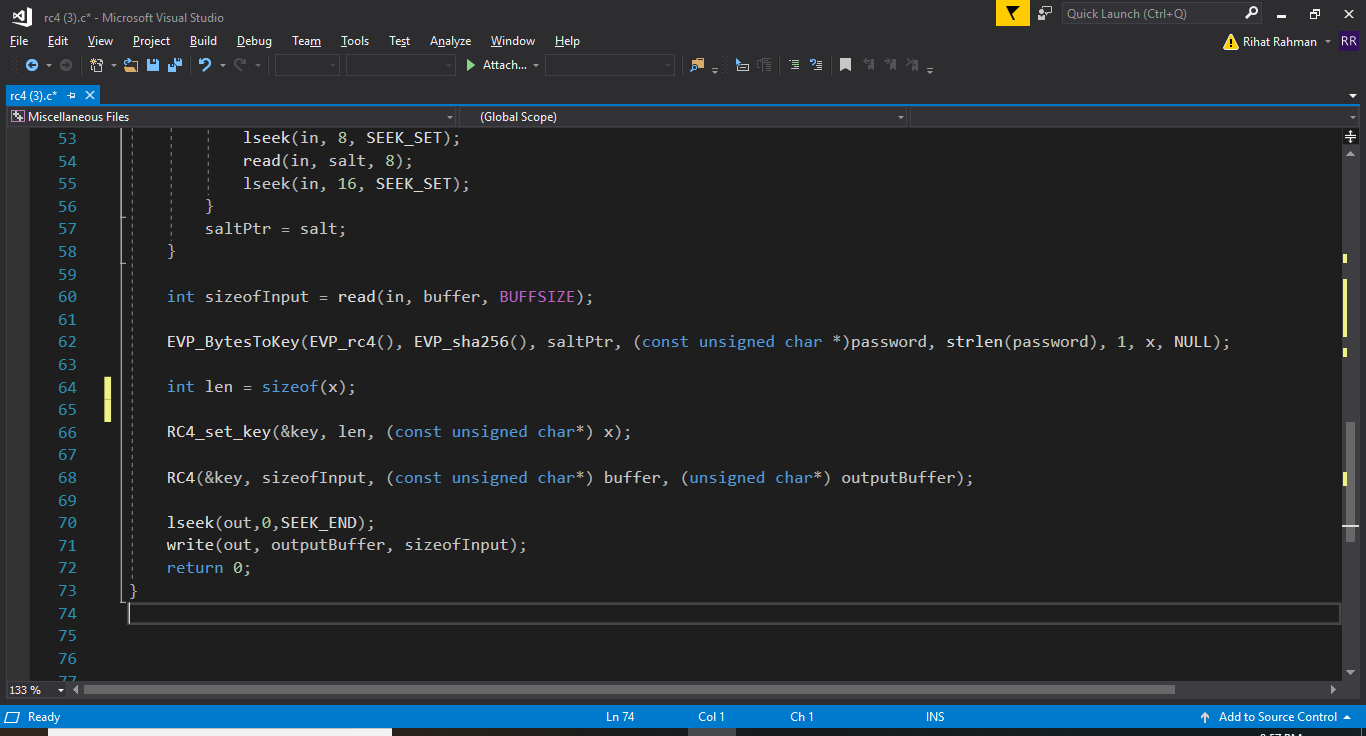
To check if the user is using salt or nosalt. After that I checked if they are trying to encrypt or decrypt the input file. If the user is trying to encrypt, I print Salted\_\_ as the first 8 characters of that file. If they are trying to decrypt, then I start reading after the first 8 characters.



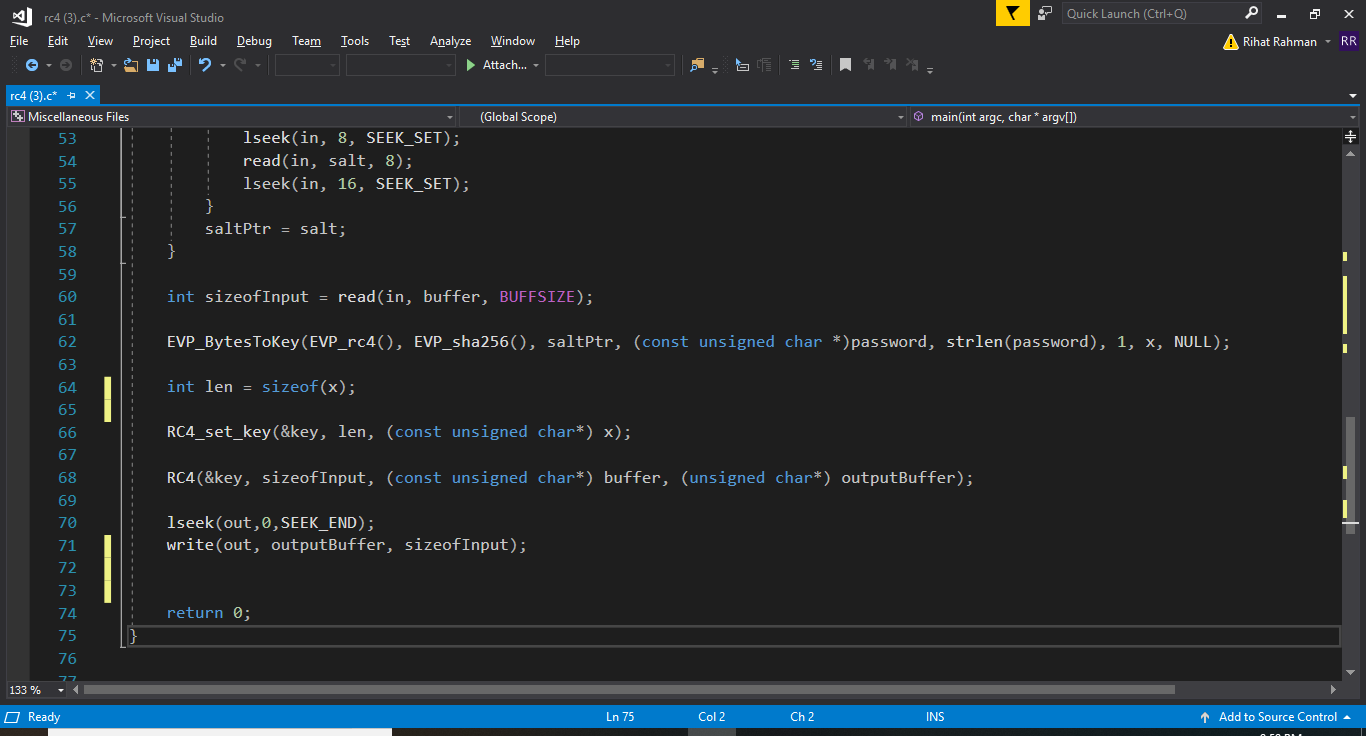
Called EVP\_BytesToKey()to obtain a key



Called RC4\_set\_key() to set up the key



Called RC4() to perform the actual encryption/decryption process



To write the encrypted/decrypted data to the output file

Command to compile

gcc -o rc4 rc4.c -lcrypto

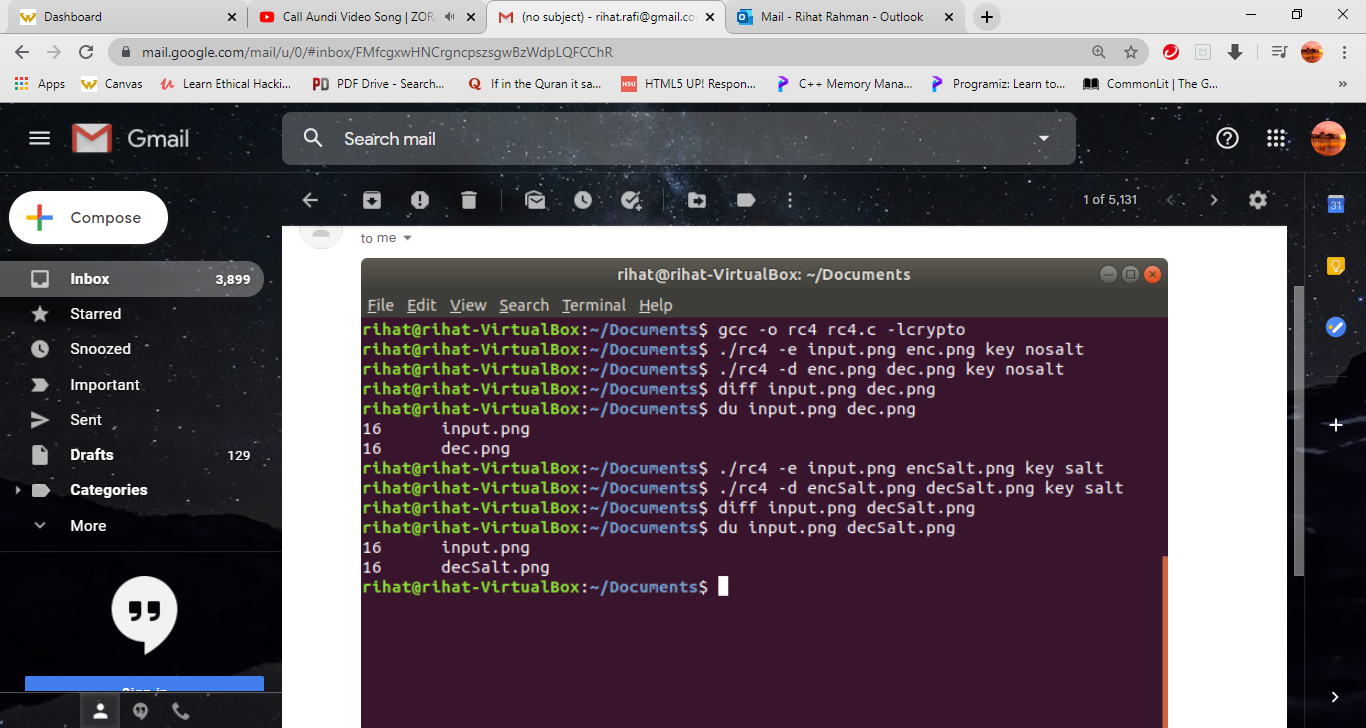
Commands to Encrypt and Decrypt using standalone RC4

./rc4 -e input.txt enc.txt key nosalt

./rc4 -d enc.txt dec.txt key nosalt

./rc -e input.txt enc.txt key salt

./rc -d enc.txt dec.txt key salt



Future Improvement

Although the class has finished, I plan to continue working this till I get encryption/decryption fully integrated into s3fs. As I currently have a better understanding of s3fs than before, I believe it would not be much difficult to accomplish that goal. I also plan to work on my standalone to make the salt part compatible with OpenSSL. I also plan to use this encryption/decryption method to make my online data secure. For that purpose, I will have to make my RC4 good enough to work with much larger files.

Summary

This was undoubtedly one of the best projects in Computer Science degree at Wayne State University from numerous perspectives. Because of the efficient study method in this class, I was able to learn how to learn by own and learn from mistakes. I started the implementation of my standalone with the thinking that I will have to create my own RC4 algorithm instead of calling the RC4 function which led me to study how RC4 encrypts and decrypts a file. I believe I can use this work more to make my salt compatible with OpenSSL. Aside from that, this class taught something that I can make use of myself. After making integration work properly, I can incorporate this project into my daily work. It is really important to ensure that the data I save online is secure and this project has the perfect capabilities for that. Finally, this was one of the best projects I’ve ever worked on. Even though I was not able to finish everything, I was able to learn a lot of stuff that will be useful in future.