Tammie Pease

CS 499

Artifact Two

Algorithms and Data Structure

This artifact was made for Secure Coding Module Five from SNHU. It’s first rendition has us take a file in which we would put some sort of input and by implementing XOR encryption, take that file and then run it through the XOR encryption. This would create an encrypted file but also the code allowed for a decrypted file to be created as well. We were then to submit the code, files, and beginning file for grading. This piece of software was used to help exemplify the points of the module and like with the former piece; this was not the main aspect of the module. It was used to demonstrate what is going on, not to give us a hard coding task.

I selected this piece for my Algorithms and Data Structure artifact for the idea that improving upon the encryption algorithm used within the program felt like the next step in the process. As like with real life, encryption and how it was implemented became more complicated and more organized over time from where it started. So, I wanted to show my ability to create new algorithms by created a new encryption method to be used for the piece of software. For the fact that XOR encryption is a type of algorithm used to create the encryption it would improve upon the algorithm and accomplish the task. So, by expanding on the algorithm and what types to use was something I approached as a way to complete this artifact’s update. In the end of the allowed time, I was able to implement choices to encrypt text on the command line or through the file. Along with this I was able to give command of what the cipher key was to the user and thus allow for different ones that could make different encryption methods. Along with this, the option of adding a Caesar encryption method was placed into the system, in which similar aspects of having it used for the file or through the command line with the number of letters shifted being chosen by the user as well. Then finally, I made a section in which you would be able to choose to do text or file of both of them combined and which one was encrypted first and then the other leading to some interesting results. It added 5 more usable outcomes than the original and it also allowed for user input to rule over the system for how much and how little they wanted to change the system. It also allowed for the implementation of showing how you do something, in this case, encryption keys and length of shifts can change what is going on. As when you do XOR first and then Caesar the symbols of XOR would create strange cases in Caesar which would not be the same and thus not decrypt the same but if you did Caesar and then XOR the decryption would work fine. Showing that the one works fine either way, but one does not like the other’s uses of symbols in the way it was implemented as it was only to work with the alphabet. But expanding the project to five more outcomes and also giving huge user input was something that the first project couldn’t even think of doing. There is more that could be done but that is something for a different day and a different project enhancement.

This project aligns closely with outcome 03 in that one of the main aspects of 03 being “managing the trade-offs involved in design choices” was something that was felt throughout the entire update. Initially, I tried to see if I could create an encryption method using OpenSSL, however when doing more digging into it, instead of an algorithm it was going into the GitHub of the system. After downloading that it was to be pulling header files from the OpenSSL GitHub and then implementing that header into my program and using some already made algorithms. This was something I was uneasy about doing for a project of this scale and idea for the fact that it was not me making a new algorithm but using another’s. Which would remove one of the main aspects and by doing so, remove what made the encryption project what it was with XOR. Also, not using an algorithm within the project but from another system entirely which for security would make sense for something needing that scale of reliability. However, it would not be best here when the goal was to improve the algorithm and data structure of the system through my own implementation. By adding these two encryption algorithms and two different ways to use them, it felt as if it kept the original aspects of the project but increased scalability. The updates to my outcome coverage plans are that the scope of what I did was changed, for the abilities of what I can do within a time limit and what I am comfortable with doing. I was able to see the solution in which I was unable to do, and so I created a capability I was able to do while holding the same project idea I was before. Thus, making a tradeoff for what was able to be done and understanding why it needed to be done. This was not exactly the modification that I had in mind however when it comes to the outcome coverage plan it fits well with 03 of this plan.

This also covers another course outcome in 05 with which I tried to create ways in which the code to avoid the code from acting in ways that would make it hard to properly function. Such as adding the Caesar key helping show different ways of adding encryption and an extension of many ways of adding security, but some are better than others. Showing how it would not always work out as planned and testing is important. Having it as a test and showing where those points of error are allows for the faults of the system to show and be remedied before it causes major problems down the line. Overall, with this, helping plan better security for the system, create the best possible scenarios by showing everything that could happen if this route is taken, and how it all looks with encryption on this level. Most real-life situations will use stronger and more prepared encryption methods but with this program, you can take a look at how it works and that not all cases create the best security.

Lastly, it allows applying course outcome 04 in the idea of implementing a computer solution for delivering value and for industry-specific goals. Because encryption is something that is not a want but a requirement now, testing and confirming that the encryption works is incredibly important to not have a data loss. So having something that shows this level of meticulous understanding for what does and does not encrypt correctly is important to know before working with real life data to ensure the best possible result from it. The industry-specific goal was trying to help make sure that encryption is done in the best possible way that does not lose value. This can be done in a variety of ways, in which, having files that can be encrypted or inline encryption on the running of the program through the multitude of options for it preforms the necessary results for encryption. Thus, the goal in mind for improved security has been achieved through such actions.

Taking a look at everything with the plan for learning OpenSSL and how to create encryption was something that I had to study more in-depth to understand. This was the definition of biting off more than I could chew. For the fact that the OpenSSL project was on a GitHub that I felt that I had little access to, and that it was something else that had to be downloaded. I found it all sketchy for me to do without having to try to find 50 different workarounds for. It would be difficult to accomplish the tasks in mind with this program as it was taking me a day to try to dig and try to download and with little time, I couldn’t waste a day just trying to get something to possibly work. So, in the understanding that, I took the hurt ego of not being able to live up to standards I couldn’t really meet with my life and changed the scope of the project entirely. Looking up different encryption methods, I discovered one called Caesar encryption in which it would switch between different letters of the alphabet by how much the key was specified. And so, with some help for figuring out how to translate that into characters in C++, I created a system where you could specify how many times you would shift it, including spaces, and then print out the results. The goal was to then make it so it could happen in the files as well, using the same methods from the initial process but refurbishing it so that it would use an integer for how much it would move over, instead of a key like XOR did previously. Then as the last of the simpler sections, I refurbished XOR so that it would be able to have Command Line options for the system. In which, you can put a key and a string you want to be ciphered and then it creates the ciphered text. Along with this, I tried to create a system in which it balanced having info that can be placed into separate functions but also calls to avoid too much confusion. I created functions for Caesar Text and File along with XOR Text and File that can be called from the main which has a system of information in which what path of the 7 is introduced. With the last 3 being the combination paths of Caesar -> XOR, XOR -> Caesar, and then the exit path. Which from the double encryption paths each carrying text or file methods for them, making it the most complicated path to get the answers for, but showing off how the data can be placed into one another with fascinating results. This was something I was capable of doing within the time specified but showing how the algorithms work and do not work with one another was a key end result found. Along with this, making it very user-centric or file-centric depending on how the user wanted to use it. It can be changed for different uses and different files depending on whatever the user types into the code to cipher themselves later on with more understanding and knowledge of what file they are working with.

With modifying the artifact, I learned a couple of things. Doing more research into what you say you are doing, if it’s something you do not know, do not promise what you may not be able to keep with at the time of use. Also, something I touched on lightly was the idea that some security methods can go one way but not the other. As when I did XOR -> Caesar the system didn’t like that for the fact that symbols used in XOR are not accounted for in Caesar and thus create a messed-up translation and thus failing to properly decrypt. But doing Caesar into XOR created a decrypted Caesar method that did the same thing that was required of it. Showing a very important aspect of security in always ensuring it works properly before touting it as the security method to use but also to help properly test everything for full functionality. Helping ensure that it doesn’t pull a Tammie and also bite off more than it can chew for what it’s capable of. Not to mention having good coding methods such as placing them into functions for common calls will help save time for copying and pasting and so removing unnecessary bits. Also, going back to the beginning and working from there is sometimes a viable option to create something you weren’t thinking of but at the same time be proud of it. There was a lot learned and a lot more to learn still such as fulling understanding the power of GitHub, but for the time crunch I had with the work I needed to do and my abilities, I’m proud of what I made here.