Team Three Final Report

2 May, 2019

Final Report

Proposal Recap/Motivation

The idea for our project was something that we as a group, thought about for quite some time before deciding on a final idea. We wanted to implement an application that could grow and be used after the project deadline, yet useful to the community around us. The original idea was to create and algorithm that would encrypt and decrypt strings. We thought this could be turned into a communication device, securing highly confidential material. Professor Mohan thought the basis of our idea was strong, but challenged us to think bigger, how can we tie this into the community that we are a part of now? He suggested we use the emails from the Allegheny College network, creating a security blanket for Allegheny students and faculty. This is where the motivation for the idea came, as we felt we could really change the allegheny community and the networking between them.

Background

Security breaches are becoming more prevalent action in our society. To better secure emails throughout the Allegheny campus, we could implement this application across campus to secure student emails. This may seem far fetched but with the implementation that already has been presented. What has already been created towards securing emails? Similar to what we are doing with emails, disk encryption is becoming a popular source of security when it comes to high level computer data savings on hard disks. To secure this, every file on the hard disk no

matter what it is, is encrypted to allow only the owner of the disk access to the files inside. Going along with this, Google has created a two step verification. It works by notifying the user of the account if their account is being logged in on a new device. Similar with ours, only the user would login on the website and into their google, forcing users to have logins in for both.

First Implementation of EmailTester

This meeting was the first of the meetings that we have had with Professor Mohan. Present at the meeting were Nathan, Professor Mohan, and Ryan. In this meeting, Prof Mohan presented groundbreaking code implementation, EmailTester, that would spark the beginning of our project. The code ran through emails through the users Allegheny email (Professor Mohan) and stored all of the information about that email into different arrays. For example, if the user requested to read through ten emails, the email subject, body, sent from, time/date, and attachments were all broken down and separated into different arrays. To run the code there are two “jar” files and they were built by an external source much like the scanner method in java. Jar files are from external code that is not related to Java but help us bring in the outside information, emails, that we would like to use. By having these we are granted access to the Allegheny user email account. The code itself uses many concepts that we have covered in class such as breaking down files and storing strings into an array.

To run this code the user has to have an app password. An app password is created to allow the user to grant access to their account to an outside project. In our case, the app password is generated for Gmail, on either mac or Microsoft (this is dependent on user). Once this is completed, the password is hardcoded into the Java code, which then allows java access to the email. After this is applied to the code, it can simply be run with the commands “./compile.sh”

and “./run.sh EmailTester”. The output that is produced starts as the code pulling in all of the emails and breaking them down into there more basics elements. After this is finished the emails are displayed in the terminal.

Secure Email Connect App

Professor Mohan introduces a new idea through the team 3 slack channel. The new idea that Professor Mohan introduces is an app called the “Secure Email Connect App”. In his words, the app will transform the original text message file into an encrypted message using a repository of encrypting/decrypting algorithms. How it would work is, an email would be sent to an individual like a normal email. After the email is sent it would be sent through the app and then encrypted so no one can tell the message. The new message is then sent to the individual still encrypted. The individual will still use Gmail to find said message, but it will be encrypted until the user unlocks the message by logging into the “Secure Email Connect App”. Once the user is logged in the email would be decrypted, allowing for the user to read and comprehend the original message. Professor Mohan found this to be interesting as we are using Gmail to communicate daily on the campus of Allegheny, as well as the encryption and decryption of outside email messages. With several of the group members favoring this idea, it is now a new goal to reach by the end of the semester. Professor Mohan also challenged the team to think of other practical, real-world ideas that we could use with the software we are developing.

Output Struggles

The basis of this meeting was about Nate’s new source code with the user interactive messaging. Nate stated, he created a new repository, separate from the original team 3 repositories with all of the mailTester java/jar files. He created this to separate the sample code

that Professor Mohan provided and the actual code used for the final application of the project. There are two methods inside this code implementation. The main method, Email in.java, takes a very similar form to the original code that Professor Mohan provided with pulling in the Gmail messages. To actual pull in the messages, the file calls the other file, Email Data.java, to read in the emails. When the code is breaking down the email into its smaller parts, an array is created internally by using a getter and setter approach. Several arrays store different "data" from the email, for example, name, time, date, message, attachments, etc. There are few with no array creation, but instead, they call other set methods to store the information.

The main reason for this meeting was for Nate. He was struggling to output the different types of email content. A multipart email is difficult to print out, like an email with an attachment. Even comparing his code to the original code, Nate was struggling to find the output he desired. He feels his problem is that the correct part of the code is not implemented correctly into his code, allowing these errors to occur.

To answer this question Professor Mohan simply multipart emails will come inside an else if statement and print out as a multipart, if there are any attachments this will be done internally, based on the if else, it will go into each message and capture the content in each message. For this project, we are going to simply focus on text emails, without any attachments or complexity. Text messages will only be used from this point further. These features could be extended later on if we would want for another class or other kind of group project.

Professor Mohan continued by saying from this point moving forward, we need to identify some use cases for Secure mailConnect, trying to use this aspect of encryption to hide the Gmail messages that we want to send and receive. Also, how will this connect to the

Allegheny community? Professor Mohan encourages us to think about some uses to our Allegheny community, and give a reason why the audience should care about this idea, and why it should be important to them.

At the end of the meeting, the group discussed what was the next step for the project. The group decided that the next direction to move in would be working on how to send an email from the terminal using the code that has already been created (expanding Email Main/Data). Using an IMAP protocol would be a good source to find this information. It was finalized by Thursday, April 11th, we should have found and presented a solution for this problem. Professor Mohan explained, if by then the group had not found a solution, Professor Mohan would step in with his solution. One final point would be to start implementing different algorithms to encrypt/decrypt strings that we can later implement into the code.

Understanding "jar" Files

During this session, Nate was able to propose a solution to this problem. His code simply took the message in as a text converted to a string, then put into an array where it would be stored until called later. Then by logging into his account with user-oriented programs for the user to log into the server on the terminal. The information is compared through the "jar" files, and then if the password matches the username, the user is granted access to see or send emails. If a user picks to see emails, then the program that was already implemented will run. If a user selects send, they will have to enter a recipient, subject, and body to their email. This information was then stored into an array and sent through the "jar" files to match google format and send us an email. Professor Mohan was impressed with the work that Nathan has done with this part of the code.

During these team session meeting, both Ryan and Adam presented algorithms to encrypt and decrypt emails. Ryan implemented a Caesar Cipher method to complete the task. Both of these were excellent steps to bring. We were able to begin working on applying both of the algorithms to our Email Main.java. Once implemented the user simply was asked what method they wanted to use for encryption, depending on the users choice it sent the value to an if statement where it was compared to the value and ran the correct call method to use the corresponding algorithm of encryption. By the end of the class period, as a group, we were able to implement both Adam and Ryan's algorithms to the code, allowing us to now send encrypted emails through our java code.

AES Algorithm

While bouncing around ideas for the project and contributing to algorithms, Cory was able to find a third way to implement an encryption method. The method was labeled as "AES", or Secret Key. Simply the code creates a secret key using byte arrays and then, sends the key to an array storing it and allowing for it to be manipulated by changing the secret key. Cory also looked into and continues to find other types and kinds of encryption methods that can be used to compare to other algorithms that we are using in the main method. The hope is that Cory will find other algorithms that we can eventually use to create more options, but also to give a bigger sample size if we decide to test the speed of each one. Along with furthering his research in AES, Cory is researching other algorithms just in case we would need a fourth option.

Website Based Material

An idea that was presented by Adam was an idea of allowing our program to be based online where users could login and have an app password created for them to allow them access

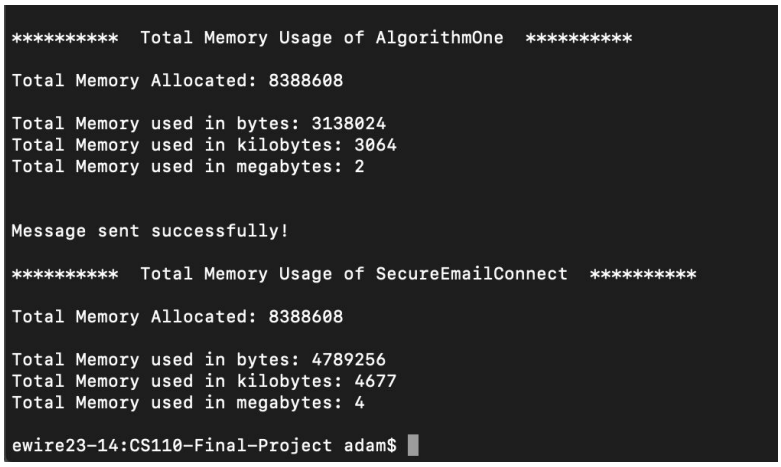
to our application. The website was made as a model where in the future of the project could be connected to. This would also be linked to the large database that would be created to store all user data and emails. The website would act as a portal for users, much like gmail is used now, with logins and security checks. The model was to show where the application could be taken and what direction it could go in. Attached below is the link to the website

https://secureemailconnect.netlify.com

Efficiency Test

In Adam's part in secure email connect was to test the efficiency of the different approaches taken to encrypt and un-encrypt messages. For each of the different methods used to encrypt the messages, he had to place the methods used within a test to figure out the amount of memory, a good measurement for the amount of data used by each of the different encryption methods. To figure out the amount of memory used, he used a function to get the amount of available memory dedicated to the program executing. This was typically around 20 MB. From this original allocated memory, the total memory used could be calculated by using the amount of memory used during the calling of the method by placing a get available memory within the encryption method. Once you have the two amounts of memory used, the used memory can be calculated by taking the total amount of memory allocated to it and subtracting the amount of available memory within the encryption method. This gives us the amount of memory used by encryption or un-encryption methods. To help understand this idea, a picture of one of the runtime tests is presented below. It tells the total memory allocated. It also breaks each one down into bytes, kilobytes, and megabytes. The algorithm that took the most memory was

CaesarCipher with taking up four megabytes. Algorithms one only took up two megabytes and The algorithm three (AES) had a total memory of three megabytes.



Also, he created a simple site that given more time we could use to create a mobile platform. To create this mobile application that could take the terminal output and place into an interactive web page it would require being able to serve the app to a server that can handle the backend. It was considered using Amazon AWS, or buying server space that would be able to run a java servlet, but this was not feasible in the time that we had currently.

In the future, it would be good to expand upon the encryption of string messages to be able to handle different types of file types to that of images, videos, applications, and more. This would require an addition to our current program, and the ability to convert these file formats to something that could be encrypted. Given more time and experience it would be great to expand with these additions.

Nate also increased his work in the project by completing the code. Form the progress report ended, Nate was able to implement all the code need to be able to encrypt every single algorithm that we had used. With encryption being the key point in the project, we also needed a decryption method. Nate was able to implement the decryption method. To decrypt the first

method, this was easy as no key was needed to decrypt the message. In this meeting as we started to go into the email details, Professor Mohan had found a small error that may be to a common user of Gmail would not know or recognize but for some, this could seem like a huge security threat.

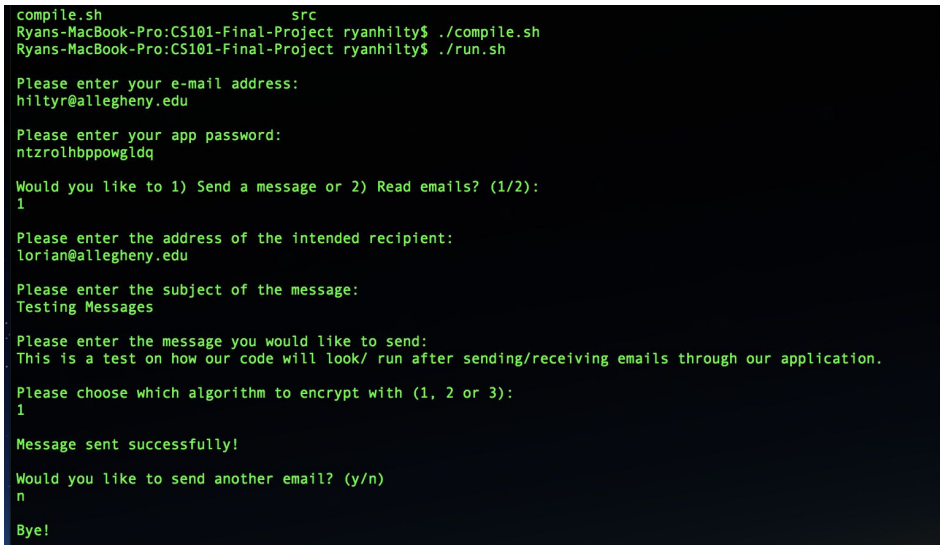
For the next two algorithms, Nate had to create a key that was stored in its own array. To hide this key, Nate sent the key with the desired decryption method through the email description. The description is hidden from the naked eye of a user but logging into the details of the emails, the key could be easily found. This was problematic as hackers or users could find this and decrypt the messages that were intended to be for the user Professor Mohan explained. Now we realize that we are not sending FBI confidential information through our Allegheny emails but the idea of this program was to keep emails confidential to the user. To fix this problem, Professor Mohan suggested that we create a database where all user entries and messages would be stored. Now, this did not seem feasible to us at first but with help from Professor Mohan, we felt that it could be doable. Although, before we dove into a problem with a semi-complicated solution, we decided to simply encrypt the keys to provide a temporary solution to this problem. These keys were encrypted and stored into their arrays until they were called on by the corresponding decryption method, either CaesarCipher or AES.

For the CaserCipher method, the method needed a key to decrypt to tell the decryption method how many letters to move over, back to the original. As described in the last paragraph the key does not need a key, because it is encrypted by algorithm one. To not create an infinite loop of keys we can only use algorithm one to encrypt the key. If a key was created for each key, there would be an infinite amount of keys created, never allowing for full decryption of the

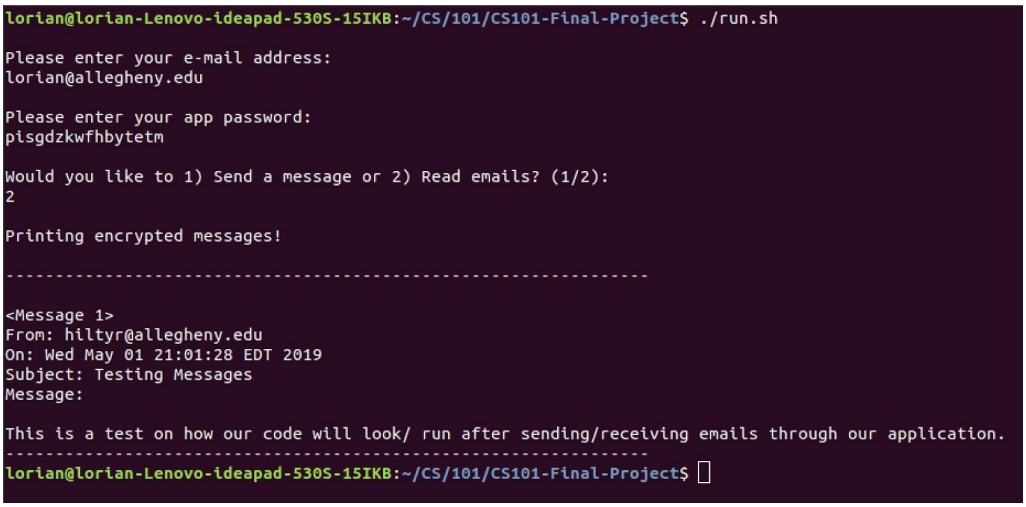
original message. Similar to this Nate implemented a key for the third algorithm of AES. This method converted the string into bytes where it was stored as zeros and ones. A key was needed to tell how many bytes there were and how to convert the bytes back to the correct string. They both use algorithm one to encrypt the key.

Final Output

Sending an Email:



Receiving Emails:



The screenshots above demonstrate how our code runs through the terminal. The first screenshot is from Ryan's Terminal of him sending a message to Nate. The second terminal is a screenshot of Nate reading in the email that was sent from Ryan. This demonstrates how the code runs through the terminal and what it all performs.

Conclusion

Throughout this project we ran into several small challenges and setbacks, but through the help of Professor Mohan and the work of our group, we were able to overcome these obstacles and create a program that we all are proud to be a part of. To highlight one of our larger problems was intaking characters as a string without extra characters. This simply was allowing for too many characters being added to the string by the substring method. To fix this problem, a simple subtraction implementation was added, subtracting two from the substring, taking away the extra characters. Our project, as a whole, could be used to revolutionize Allegheny networking through emails as data breaches increase throughout our society. One thing that all group members could be proud of, was the fact that we created a basis for an application that could grow into something that could change the Allegheny Advantage forever.

Works Cited

https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/life.html

https://support.google.com/accounts/answer/1085463?hl=en

https://docs.oracle.com/javase/7/docs/api/