Titan Newman

Security Algorithms

Professor Kippins

4/17/2022

Milestone Paper

Abstract

For an encryption project, I was tasked with creating an encryption application. I chose to create a Windows executable that takes in a user's input and proceeds to either encrypt or decrypt a file. The file is encrypted through an AES encryption cycle and the user keeps their own password to encrypt or decrypt the files. This allows the program to stay as the 'middleman' and doesn't require it to store passwords securely. Through testing, I concluded that using Java's extensive AES libraries was the better route to take (compared to my own AES encryption cycle). The code will be (as it is not finished yet) exported through a .jar file and will allow one time use (once used, it closes itself).

Introduction

My task for this project was to produce a project that works on either a Desktop, an Android device, or an iOS device. This program/project must connect/touch on some security encryption method and allow for user interactions with it. I decided to that my project will be a Microsoft Windows executable application that will take in a file and encrypt it with AES (128 bits). The application will not implement authentication as the files will not be leaving the local computer, it will randomly create a system key (i.e. a symmetric key), and it will enable the file to be decrypted based on a key given by the user. This means that the application will take in a set file, apply the AES encryption standard, and save it. We can then choose to either decrypt the file (based on its key) or leave it as is. There are already applications around that do this, however this application will use baseline Java code and will not only encrypt/decrypt files for users, but also allows myself to dive into Java's UI functionality and use already existing AES libraries (which allowed me to compare my own AES functions from the labs compared to already implemented and in-use libraries). The rest of this paper goes over how far I have gone in my project, and how I have approached this task. The paper will dive into some related work, the methodology behind the creation of the project, any testing (NOT included in this paper), and a brief conclusion (NOT included in this paper).

Related Work

There are a lot of encryption programs around, a lot of them have very similar traits. A lot of these applications not only cost money, but also allow for cloud back-up. Some applications that I based my project off include: NordLocker, Steganos, AxCrypt, CryptoForge, and Cypherix SecureIT. All these applications have their own pros and cons, however one that repeats itself is the fact they save the file's encryption password/passphrases. This means that anyone who can gain access to the application can also gain access to the encrypted file's keys. This was a problem that I didn't want for my application, which lead me to only allowing the application to take in a password once while running. This means that the passwords aren't saved and keeps the password's safely out of the application's responsibilities. I am unsure of the languages used by the related software, however most of them use widely accepted encryption libraries which helps to keep them secure. These libraries allow a magnitude of tests to be conducted (which tests for encryption errors), but also allows these encryption standards to become widely accessible. As these libraries are used frequently, I decided to incorporate them into my project to allow for the latest and strongest versions of AES encryption.

Methodology

For this project, I decided to work from the ground up, but stuck to Java as the language as I could fall back onto my AES encryption function if the libraries didn't work out. Java has an extensive library of import which allows for faster and cleaner code to be produced. I started of with creating the UI for the application, where ease of navigation for the user was the focus. I stuck to using Java's 'javax.swing' import which holds functionality for both free user selection (selecting files) and for user inputs (input passwords and selecting what needs to be done [encryption/decryption]). I then made sure that I was able to properly read a file's data and save it locally. The next stage was to run through the encryption/decryption process for the data in a file. Based on the user's input, the application would call the necessary functions to properly incorporate the AES. For this, I used the imports "javax.crypto" and "java.security.spec" to successfully encrypt/decrypt files (the data inside of the files). The functions connected to these imports use an AES-256 bit encryption cycle. At the end of the project, I conducted test cases to make sure that encryption/decryption cycles completed correctly (as per the next section).

Testing

Not conducted yet.

Conclusion

To be written after completion of project.

Bibliography