Jillian Laliberte
CMPT Milestone Project

Abstract

The program developed in this project aims to accept String inputs for Website and Password values from a user and then stores an encrypted version of the password in a hashmap in a separate location. In order to retrieve the values of the password the user must enter a passcode designated by the programmer. Encryption is done by means of a symmetrical algorithm which examines the ASC value of each letter in the string and alters it prior to storage.

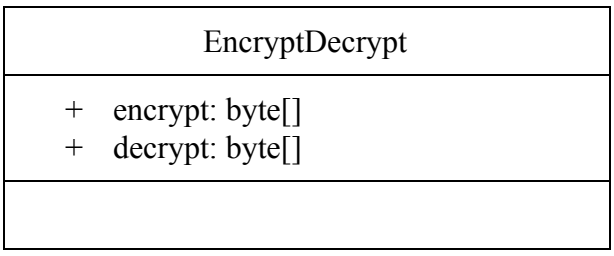
Introduction

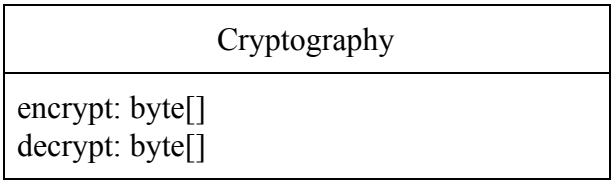
This code aims to create a convenient and secure location for the storage and retrieval of passwords for a variety of websites. The relevance of this project lies within increased use of technology within the past decade. Not only does this increase demand for the creation of passwords, it has also consequently led to decreased utilization of previous storage methods. Passwords and other pertinent information are now commonly stored in digital form, rather than written down. Further, a secure method of digital storage is a relevant and significant topic of interest. Most commonly, sensitive information is protected in technology via cryptography. Various methods of encryption and decryption are utilized and studied within the realm of programming, however this specific program secures private information via an algorithm which stores passwords as an altered version of their ASC value. The encryption of passwords prior to storage is an integral mechanism of preventing sensitive information from being compromised.

System Description

The interface “Cryptography” declares two arrays of bytes which will be utilized by the EncryptDecrypt class to alter strings inputted by a user. Byte arrays were selected over other numerical data types due to the ease by which other data types are converted to bytes. Within the EncryptDecrypt class, a secondary array of the same length of the original is created in order to store the encrypted version of the password. Additionally, this class contains the algorithm which alters the value of the inputted string for storage. The algorithm examines the ASC value of the character, if said value is even a one is added. Conversely, if the value is odd, one is subtracted from the value. The mirror image of this algorithm is utilized by the array ‘decrypt’, which can be used to return the string to its original ASC value.

The class UserInput acts as the interaction point for users. Here, a HashMap called ‘storage’ is declared in order to link inputted website titles to their respective encrypted passwords. In order to access previously entered and stored values, the user must input a four digit code to unlock the safe. Then, the decrypt function of class EncryptDecrypt is utilized to return the original form of the inputted password and its associated website.





Requirements

This project addresses one of the most basic aspects of cyber security, password protection. By conveniently storing passwords in an encrypted and locked form, users are able to safeguard information they likely use for more than one website without having to memorize all of their passwords. Utilization of an encryption algorithm that alters the stored ASC values of a given character provides increased security as it limits a potential hacker's ability to understand the stored encrypted version of the password.

Literature Survey

Evaluation of literature regarding mechanisms of password protection, the key to safety seems to be never storing passwords as plain text. Regardless of the numeric lock on the safe, passwords can be very easily hacked if not encrypted. Further, the degree of security provided is directly related to how difficult it is to revert a string to its original form. Certain mechanisms of protection suggest using hashing algorithms rather than encryption. This provides added security because hashing is a one way process as opposed to the two way path of encryption/decryption. Further literature suggests the utilization of a randomly generated symmetric or asymmetric keys which guide encryption, rather than a hardcoded cipher that can be easily accessed.

User Manual

Users select an action to enact on the password hashmap, their options being to add or view prior passwords. The user must input a 4 digit numeric key in order to carry out any function. From there, users must input the name of the website and its respective password if they are choosing to store a new input. Once websites and passwords have been entered into the

hashmap, they can be accessed via the website title which acts as the key to finding the respective password. If the incorrect master key is enetered at any point, a print statement which is executed to inform the user as such.

Conclusion

Convenience and speed are qualities of utmost priority for technology today. If a program runs slowly or is difficult to understand, users will likely go elsewhere to find more streamlined versions with the same functionality. Therefore, the simplicity of the program improves its ability to be marketed to users. The importance of cybersecurity has been highlighted throughout this discussion, especially in this age of growing technology. Thus various mechanisms of protection and encryption are an increasingly relevant area of study. Furthermore, a simple and secure mechanism of information storage and retrieval is a useful and sought after development provided within his system.

Bibliography

Sundar, Veera. (2010). Storing passwords in Java web application. Dzone.

https://dzone.com/articles/storing-passwords-java-web