Cameron Ulrich

Total Image Encrypt

MSCS 630 Project Final Writeup

Abstract -- This final write up will discuss the implementation of a project that uses AES (Advanced Encryption Standard) to encrypt a text with a key, and embed the text into an image. T

1. Introduction

The idea for this project couldn't have come at a better time. With Coronavirus keeping a majority home in quarantine, people need to work online. As a result, many important and sensitive documents and images need to be sent across the internet to other employees. The AES algorithm allows for both users to have a key beforehand and so the text cannot be seen until decrypted.

2. Background and Related Work

Many other researchers and past students in this class have done similar projects with image encryption and also text encryption, but I haven't seen both be done together. Unfortunately, due to time constraints I was not able to succeed in figuring out how to successfully encrypt an image.

3. Methodology

Currently, the project is using React to implement everything as a webapp. I have implemented an AES encryption algorithm on an encryption page. I used an AES library called crypto-js/aes [2]. The encryption page can be divided into 3 steps. The first is to type in the text you want to be encrypted. The second is to type in a 16 or 32 digit key. The key is the most important piece of information and thus should be kept as secret as it can be. The key will be given to all the users so they can decrypt the image. Finally, once that is all typed in, an image input box will appear for the user to input any image of their choice. The image will appear in the bottom left corner of their screen to which they can right click and save the image. This is the newly encrypted image. On the decrypt tab, it is laid out very similarly.

The user starts this time by uploading the encrypted image they were given. After uploaded, a secret text is painted onto a canvas below the encrypted text input box. This is to deter hackers as the text isn't actually anywhere to copy, it is physically painted onto a canvas. The user manually types in the encrypted text and then types in the key that they should have ahead of time. Then they click the decrypt text button and the decrypted text appears next to the secret message label.

4. Experiments

The tests performed is making sure that the text is successfully encrypted and that the user with the same key can decrypt that message. The next test is having the text successfully be embedded into an image. Then the next test would be saving the image, and seeing if it can be decrypted. The next test would be testing the text from the image and decrypting it with the key and without. Without the correct key it results in random symbols or characters.

5. Discussion and Analysis

Building with React has gone very smoothly. I was able to work with react last semester for my capping project, so I repicked it back up. Additionally, the encryption and decryption for the text with a key went very smoothly. The image encryption however, was a much tougher time to research and figure out. I tried many different ways and tutorials online, but with the time constraint I was unable to implement it in my project. I can argue, however, it may be better not having the image encrypted or jumbled because it would attract hackers to test and try to decrypt the image. While on the other hand, if the image looks like a normal image, it won't gain any attention by any hackers.

6. Conclusion

Overall, the project was very interesting and although at many points it was challenging, I enjoyed it. I may keep searching for ways to implement image encryption into the web app, but I am very happy with how it turned out.

7. References/Bibliography

[1]. P. Radhadevi, P. Kalpana, "Secure Image Encryption using AES", P. RADHADEVI* et al, Volume: 1 Issue: 2, ISSN: 2319-1163, page 115-117.

[2].https://www.npmjs.com/package/cry
pto-js