Lock My Text: File Encryption and Decryption Application with AES-256-CTR on AWS EC2

https://github.com/mraposka/Lock-My-Text

Abdulkadir Can Kinsiz
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
E-Mail:kadircankinsiz@gmail.com

***Abstract*—Lock My Text is a simple file encryption and decryption application that leverages the AES-256-CTR algorithm. This project provides a secure method to encrypt and decrypt files using a password. Built with Node.js and containerized using Docker, the application features a web-based interface for easy user interaction. Users can upload a file, input a password, and download either the encrypted or decrypted version of the file. The use of Docker ensures that the application is easily portable and scalable across various environments, specifically in cloud-based environments like AWS EC2.**

***Keywords*—AES-256-CTR, file encryption, file decryption, Docker, Node.js, AWS EC2, security, web application, containerization**

I. Introduction

In today's digital age, data security is a major concern for individuals and organizations alike. With the increasing use of cloud storage and file-sharing platforms, the need for secure file encryption and decryption methods is more crucial than ever. This report explores the development of **Lock My Text**, a simple yet efficient file encryption and decryption application built using Node.js and Docker, which is deployed on an AWS EC2 Linux server. The application allows users to securely encrypt and decrypt files using the AES-256-CTR encryption algorithm, widely recognized for its robust security.

The goal of this project is to provide an easy-to-use solution for individuals needing to protect sensitive files. By utilizing Docker, the application can run on any machine with minimal setup, making it highly portable and scalable. The use of AWS EC2 provides an additional level of flexibility, allowing the application to be deployed in the cloud for global access. Users can interact with the application through a simple web interface, where they can upload files, apply encryption or decryption, and download the processed files. This report covers the design, ease of use, deployment process on AWS EC2, and potential applications of **Lock My Text**.

II. Ease of Use

Lock My Text was designed with user-friendliness in mind. The web-based interface allows users to quickly upload their files, input a password, and receive the processed file without requiring any technical expertise. The simplicity of the interface is a key feature, ensuring that anyone, regardless of their technical background, can easily use the application.

The web interface consists of two main pages:

1. **Lock**: Users upload a file, enter a password, and the file is encrypted using the AES-256-CTR algorithm.
2. **Unlock**: Users upload an encrypted file (.enc), enter the password, and the original file is decrypted and made available for download.

By utilizing Docker, the application can be deployed in any environment with minimal setup. This reduces the complexity of installation and ensures that the application can be used in various production and testing environments. The application was deployed on an AWS EC2 Linux server, ensuring its availability for remote access. The EC2 instance provides a stable, scalable environment for the application, making it suitable for both personal and enterprise use.

Overall, the ease of use and setup make **Lock My Text** a versatile and accessible tool for anyone in need of secure file handling.

III. Results

The **Lock My Text** application was successfully developed and deployed as a Docker container. The encryption and decryption processes worked as expected, with files being securely encrypted and decrypted using the AES-256-CTR algorithm. The web interface provided a simple and intuitive way for users to interact with the system, and the system handled multiple file formats effectively.

Performance tests showed that the application handled file sizes up to several megabytes without significant delays. The encryption and decryption operations were completed within an acceptable time frame, with no noticeable performance degradation for typical use cases. The Docker containerization allowed the application to be easily deployed and run on different machines, ensuring portability and scalability.

The deployment on an **AWS EC2 Linux server** proved to be efficient and seamless. The EC2 instance provided the required computing resources and networking capabilities to run the application smoothly. The web interface was accessible globally, and users could interact with the application remotely, making it ideal for cloud-based deployments. In terms of security, the use of AES-256-CTR ensured that the encryption process was robust against potential attacks. However, the security of the application is also dependent on the strength of the user-provided password.

Acknowledgment

A special thanks to **Metehan Güney** for his continuous support and insights throughout the development process. I also extend my thanks to the open-source community for providing tools and libraries such as Node.js, Docker, and the AES-256-CTR encryption algorithm, which were essential in the development of this application.

References

[1] *Node.js Foundation. (n.d.). Node.js Documentation.* https://nodejs.org/docs/latest/api/

[2] Bootstrap. (2024). *Getting started - Introduction*. Retrieved December 5, 2024, from https://getbootstrap.com/docs/5.3/getting-started/introduction/

[3] Docker, Inc. (n.d.). *Docker Documentation*. https://docs.docker.com/

[4] Amazon Web Services, Inc. (n.d.). *Amazon EC2 Documentation*. https://docs.aws.amazon.com/ec2/

[5] Stack Overflow. (n.d.). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. Retrieved December 5, 2024, from https://stackoverflow.com/