VELLORE INSTITUTE OF TECHNOLOGY

School of Computer Science and Engineering

Fall semester 2020-21

CSE2005 Operating Systems

J Component Report

on

CryptSend: Integrated Encryption Scheme Based File Sharing System

By

Pranav Sinha - 19BCE0838
Bangoria Jeet – 19BCE0874
Chinmay Sai Vemuri - 19BCE0901
Pahchan Garg - 19BCE0910
Kanani Harsh – 19BCE2562

SUBMITTED TO

DR. VIJAYA KUMAR K

ACKNOWLEDGEMENT :

Our special thanks go to one and only **Prof. Vijaya Kumar** who supported us throughout our journey. This project consumed huge amount of work, research and dedication.

I would also like to acknowledge VIT for giving the candidates an opportunity to carry out our studies at the institute.

Abstract :

CryptSend is a service that allows users to send and receive sensitive messages, files, and documents securely, with end-to-end encryption, using just their web browsers. Encryption takes place in the sender's web browser, and decryption takes place in the recipient's web browser, via client-side JavaScript running in both users' web browsers. Only encrypted information is sent CryptSend’s servers. Unencrypted information and the keys used to encrypt/decrypt this information never leave the sender's web browser or the recipient's web browser. Senders may send information to a recipient through the service, without registering for the service, simply by knowing the recipient’s username on the service.

Introduction :

It is common for individuals to have a need to exchange information in a secure and confidential manner. For example, accountants, attorneys, and medical professionals frequently have a need to send and receive confidential information to and from their clients. Because standard SMTP email does not provide a means for encrypting information from the sender's end to the recipient's end of an email transmission, users have sought solutions to use instead of SMTP email, or in conjunction with SMTP email, for exchanging information confidentially and securely.

Not all people are familiar with message encryption and its techniques. People who are familiar know that it isn't a trivial task and chances of wrong implementation are high. As the world wide web evolved, cloud-based file sharing services (such as those offered by Citrix® and Dropbox®) have emerged, which claim to allow users to share files securely. These services typically allow users to share files via the following procedure: First, the service allows a sender to upload a file using their web browser to the service's server through an SSL/TLS connection. The SSL/TLS connection ensures that the file is encrypted while in transit from the sender's web browser to the service's server. Then, upon receiving the file at the service's server, the file is immediately encrypted by the service using a symmetric encryption algorithm such as AES, so that the file is encrypted while at rest while stored on the service's server. The AES encryption key is also stored by the service. Later, when the file is requested by the recipient, the service uses the encryption key to decrypt the file. Then, the service allows the recipient to download the file using their web browser from the service's server through an SSL/TLS connection. The SSL/TLS connection ensures that the file is encrypted while in transit from the service's server to the recipient's web browser.

However, astute observers will quickly notice that this is not an end-to-end encryption solution. There are points in time when the file is unencrypted while in the service's possession – specifically immediately after the sender uploads the file, and immediately before the recipient downloads the file. Moreover, even though files sent through the service are encrypted while stored by the service, the service has the ability to decrypt these files at any time, because the service also has possession of the encryption keys. Notwithstanding, these services are sometimes seen as more convenient than other solutions, because they can be accessed using a web browser, eliminating the need for the sender and/or recipient to install special software.

CryptSend is an attempt to achieve true end-to-end encryption by implementing browser based message encryption system. This service is available to everyone with internet access and requires almost no technical background for its usage. In addition, senders are able to send files and messages to recipients without the need for senders to be setup on the service, simply by knowing a unique username, address or URL for a recipient. Its main aim is to enable secure messaging for everyone.

Implementation Theory :

Our system looks to deter man-in-the-middle attacks. Encryption takes place securely at the client-side in the client's web browser. For the encryption to take place, a secret key file containing the private key among other things is necessary. This secret key file is generated during registration and needs to be stored on the device. When logging in, this secret key file is enough for authentication of the client. The secret key file is encrypted using a pass-phrase entered during the registration process. This pass-phrase has to be re-entered during login process to unlock the secret key file.

To send a file to another user, the recipient's username is required. The system fetches the public-keys of the recipient which are then used in the encryption process. After the encryption process takes place in the browser, the file is sent to the server-side wherein it is stored in a database. The file is in encrypted form when it is stored in the database and hence no-one, not even the server-side developers can open the file. To receive a file, the client simply has to login and the received files will be visible.

Certificates proving the authenticity of the server, recipient and client is necessary but due to the complexity involved and the time constraint we were under, we couldn't implement it into the system. This will be implemented in due time.

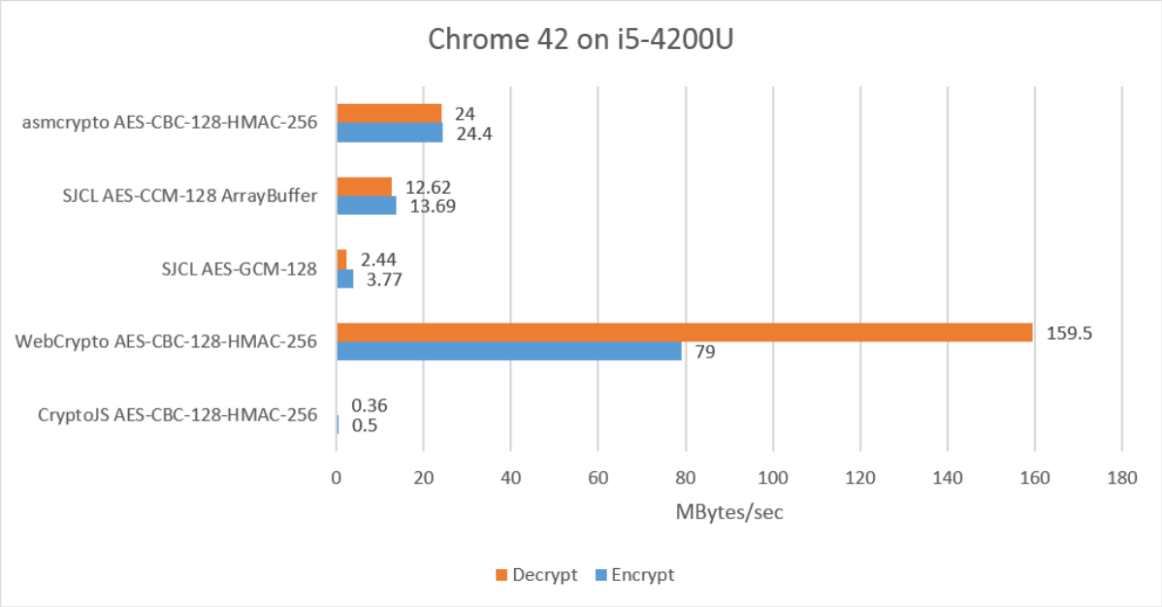
For the encryption process, we utilized the following cryptography primitives: Elliptic Curve Diffie-Hellman Key Exchange Protocol (ECDHE), 256-Bit AES-GCM symmetric key encryption and decryption, PBKDF2 password based key derivation function.

The ECIES standard combines ECC-based asymmetric cryptography with symmetric ciphers to provide data encryption by EC private key and decryption by the corresponding EC public key. The ECIES encryption scheme uses ECC cryptography (public key cryptosystem) + key-derivation function (KDF) + symmetric encryption algorithm + MAC algorithm. Integrated Encryption Scheme (IES) is a hybrid encryption scheme which provides semantic security against an adversary who is allowed to use chosen-plaintext and chosen-ciphertext attacks.

Implementation Details :

The website that we created utilizes WebCrypto API at its core. This API is recommended by the W3 Consortium for cryptography purposes. The main reason we used this API instead of standard libraries is due to the following reasons:

- The javascript libraries for cryptography are generally very slow as compared to WebCrypto API.



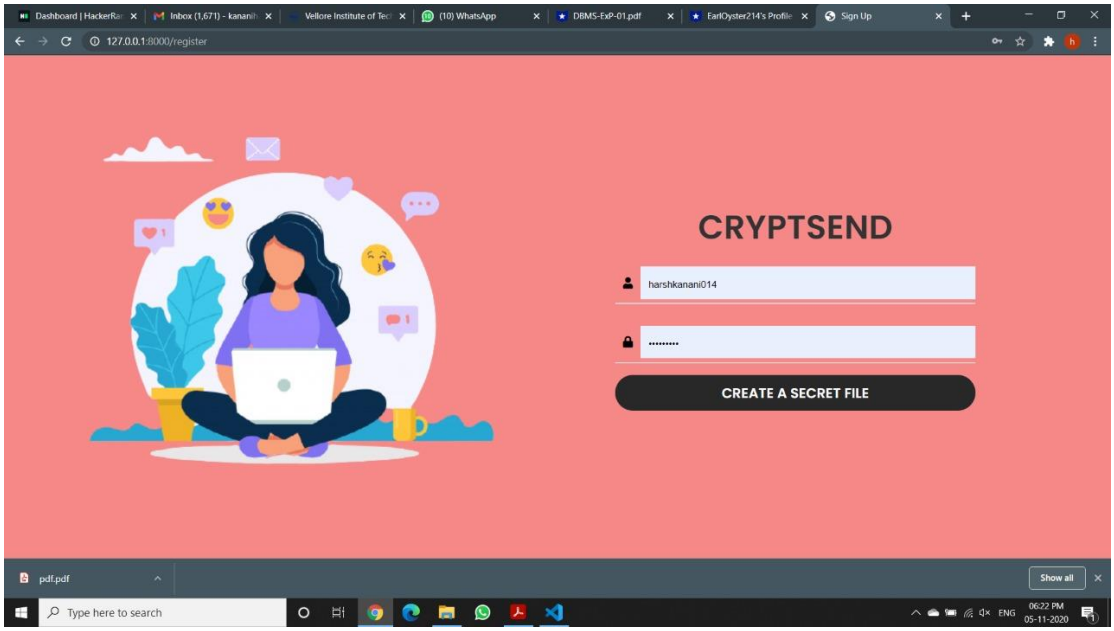
Source: **https://medium.com/@encryb/comparing-performance-of-javascript-cryptography-libraries-42fb138116f3**

As the picture above shows, WebCrypto API is much faster than other javascript cryptography libraries. This is due to the fact that the API uses low-level interface and hardware resources exceptionally well.

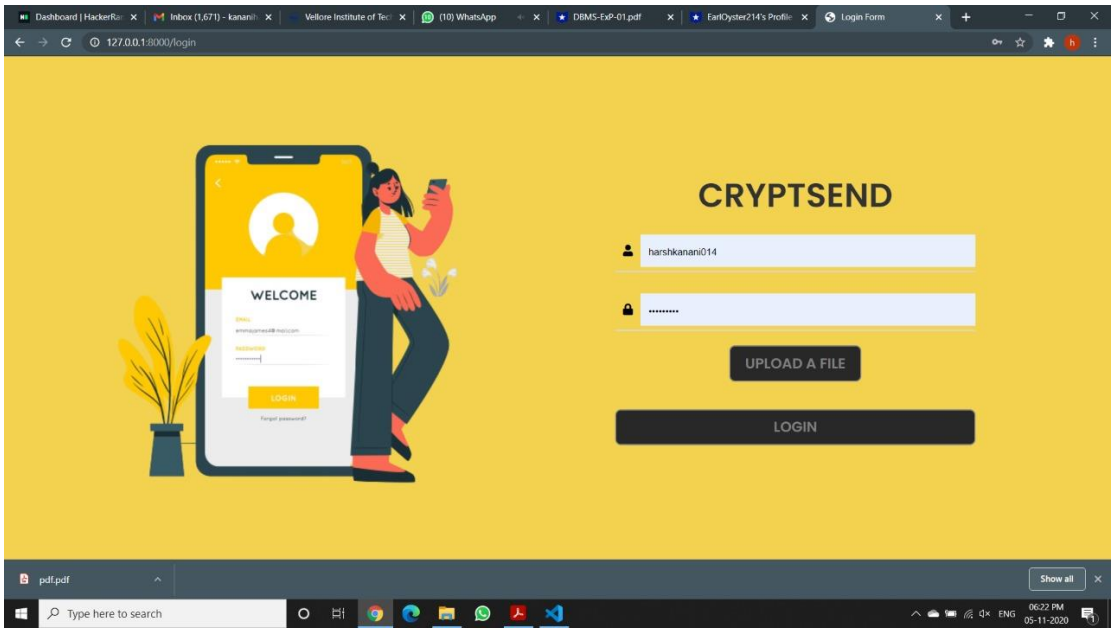
- WebCrypto API is generally easier to use if one knows exactly what they are doing. The documentation of the functions provided by the API is

  The server-side is managed by Django framework and PostgreSQL. The communication between the client-side and server-side is done using the Fetch API. This API allows us to easily post and get the data to and from the server-side using JSON requests.

Pictures :



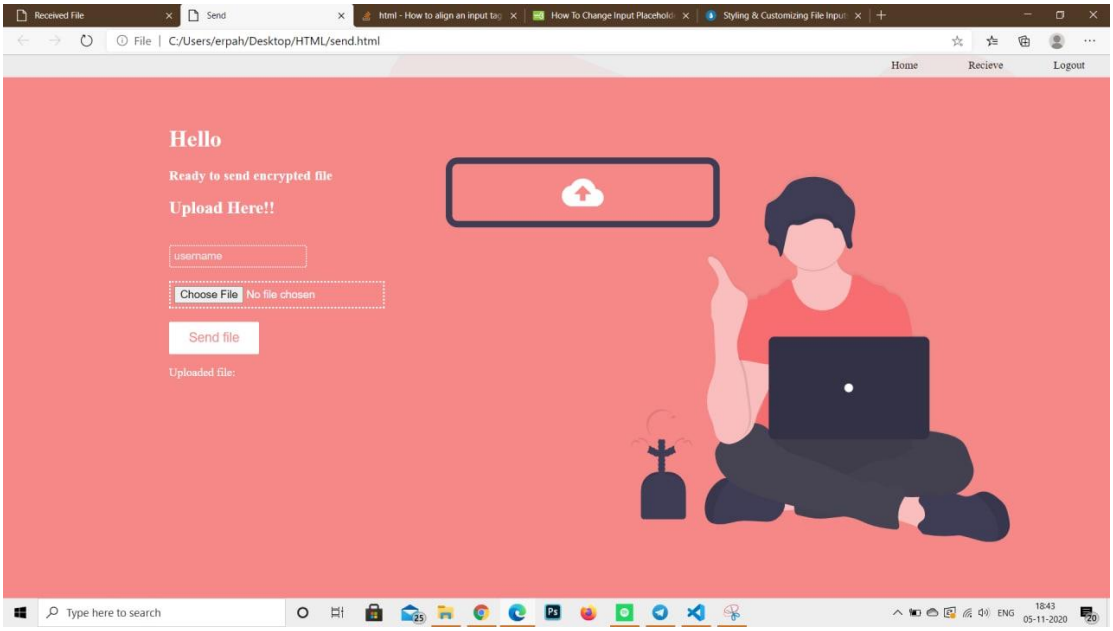
Registration Page



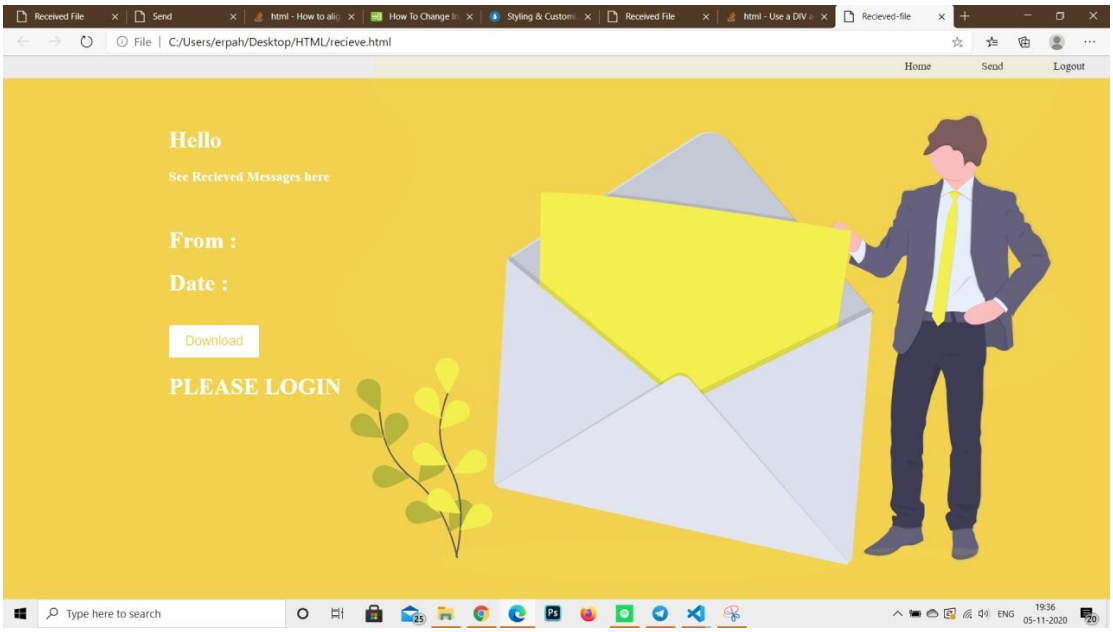
Login Page

Home Page

Authenticated Home Page Page



Send Page



Receive Page

Github Repository link:
https://github.com/harshkanani014/Web-based-File-sharing-using-using-integrated-encryption