[1]

Encrypted Instant Messaging

Lazo. David, Mosquera. Raúl, y Vilchez. Osman. Estudiantes de UTEC Computer Science

I. INTRODUCCIÓN

El Encrypted Instant Message (EIM, por sus siglas en inglés), la mensajería instantánea es un servicio de internet que nos permite tener una comunicación en tiempo real con otros usuarios la cual compartimos la misma aplicación de mensajería instantánea. Ahora, EIM generalmente se clasifica como públicas o empresariales. Tenemos algunos ejemplos sobre mensajería instantánea como AOL, Yahoo Messenger, Microsoft.NET Messenger. Asimismo, se toma en cuenta que el acceso al servidor de mensajería está restringido y se toman precauciones de seguridad, con el cifrado, para poder proteger la red empresarial.

Entre las diferentes formas de implementar un EIM, uno de los más seguros es el end-to-end, que es el *state of art* de los métodos en apps como WhatsApp o Telegram. Ello es posible mediante tres enfoques criptográficos: simétrico (si se tiene un canal establecido o se conoce el secreto compartido antes de la comunicación), asimétrico (para establecer el secreto compartido) e híbrido, el cuál junta lo mejor de ambos a costo de una mayor complejidad de implementación.

Presentado el contexto, el presente documento, es el informe final de un chat end-to-end simétrico.

II. DESARROLLO

A. *Características*

La aplicación cuenta con diferentes métodos usados, tales como:

- Responsive
- Interfaz Sencilla (UX)
- Arquitectura sencilla
- Mediación completa (autenticación, autorización e integridad)
- Diseño abierto
- Diseño modular
- Utilización de herramientas probadas (sqlAlchemy, passlib, crypto-js, SendGrid)
- Verificación de data types
- Verificación de inputs

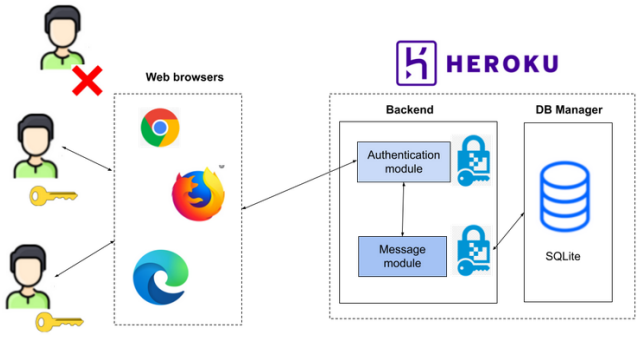
B. *Arquitectura.*

Vamos a tener una arquitectura de cliente servidor, donde a través de una página web dos diferentes usuarios podrán comunicarse de manera segura.

Se usa el lenguaje de Python con Flask (backend), con JavaScript (frontend), usamos el ORM sqlAlchemy y un GBD Sqlite.

Para este tipo de plataforma el usuario se tendría que loguear con su cuenta de correo y acceder con el PIN gracias al api de SendGrid.

A su vez al momento de hacer un clic en la casilla del mensaje de cada usuario conectado va a pedir un archivo (llave), para que pueda acceder a la visualización de diferentes mensajes.



C. *Seguridad.*

Encriptación de PIN:

Usaremos mecanismo de encriptación (simétrico), con el fin de tener una mejor validación al momento de comunicarse. El usuario tendrá que loguearse con su correo y recibirá un PIN para su verificación y tener en cuenta que con la persona que estamos hablando sea la misma persona es quien dice ser y no una suplantación.

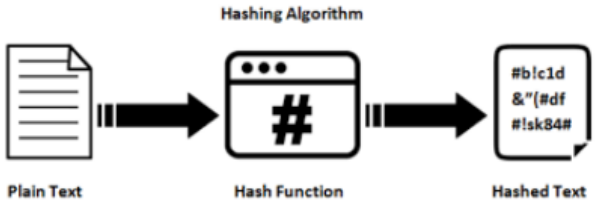
Encriptación de mensajes:

Al momento de ingresar a los mensajes de un usuario se tendrá que ingresar un texto (llave) para que el usuario pueda ver cada mensaje con el fin de asegurar la confidencialidad gracias a AES.

Encriptación de contraseñas:

Para esto usamos Hash SHA-256 al momento de hacer la encriptación de contraseña, esto nos ofrece un nivel alto de seguridad, lo que es perfecto para proteger y codificar de manera segura la información. Este algoritmo es de una dirección.

[1]



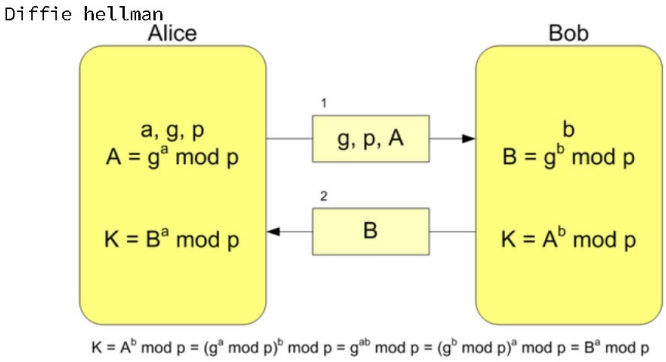
Tipos de llaves-Mensajes:

```
from passlib.hash import sha256_crypt

password = sha256_crypt.encrypt("password")
password2 = sha256_crypt.encrypt("password")

print(sha256_crypt.verify("password",
password))
```

Secreto establecido (Nice to Have)



D. Problemática.

1) Probables ataques

Si bien el password utiliza SHA-256 y es susceptible a ataques de fuerza bruta, la capa adicional de generar un PIN le da una capa extra de seguridad, para poder vulnerar dicha capa, el atacante debería tener acceso al correo.

2) Desventajas

- Usabilidad baja debido al upload de las llaves.

3) Recomendaciones

Para mejorar la usabilidad, la implementación de almacenamiento de las llaves de los mensajes en cookies, junto con la generación de llaves con Diffie-Hellman.

4) Deseables

a) Utilizar el federated login (de Google por ejemplo) para autenticarse en la aplicación.

b) Crear las llaves con Deffie-Hellman

III. CONCLUSIONES

El tema de la seguridad en aplicaciones y/o sistemas es un tema muy importante a considerar desde el inicio de la etapa del desarrollo, ello permite tomar las debidas precauciones ante las distintas amenazas que pudieran ocurrir. Referente a este trabajo implementamos una chat de mensajería end-to-end simétrico con sus alcances acatados y vistos en el curso, sobre tipos de seguridad y encriptación, no solo al momento de loguearse, sino también al momento de poder visualizar todos los mensajes entre los usuarios en tiempo real.

IV. REPOSITORIO

https://github.com/davidalejandrolazopampa/Seguirdad_EIM

V. REFERENCIAS

[1] T. Bose, "Implementing end to end encryption in your cross platform app," *DEV Community*, 10-Mar-2021. [Online]. Available: https://dev.to/ruppysuppy/implementing-end-to-end-encryption-in-your-cross-platform-app-3a2k. [Accessed: 20-Dec-2021].

[2] "Hashlib - Secure hashes and message digests," *hashlib - Secure hashes and message digests - Python 3.10.1 documentation*. [Online]. Available: https://docs.python.org/3/library/hashlib.html. [Accessed: 20-Dec-2021].