Instituto Tecnológico y de Estudios Superiores de Monterrey
Escuela de Ingeniería y Ciencias
Ingeniería en Ciencias de Datos y Matemáticas

Sistema de Encriptación: Protección de Datos Personales para casa Monarca

Equipo 6:
Valeria E. Lugo Gutiérrez - *A00830523*
Axel Quiroga Caldera - *A00832676*
Valeria M. Serna Salazar - *A01284960*
Diego Garza González - *A01721186*
Rubén Darío Castro Terrazas - *A00833945*

Uso de álgebras modernas para seguridad y criptografía
Grupo: 603

Docentes:
Dr. Luis Miguel Méndez Díaz
Dr. Daniel Otero

Monterrey, Nuevo León

11 de junio del 2024

Contents

1. Abstract

Durante este artículo se presenta una investigación detallada sobre el método óptimo de encriptación para salvaguardar la información sensible de la base de datos de ”Casa Monarca - Ayuda humanitaria al migrante A.B.P.”, un refugio para migrantes en Monterrey. Se selecciona el estándar AES, debido a su eficiencia computacional y seguridad robusta, y se elige el modo de cifrado AES-GCM por su capacidad de proporcionar confidencialidad y autenticidad. La implementación se realiza utilizando una base de datos relacional en la nube y una interfaz de usuario desarrollada en HTML, CSS y JavaScript con su Back-End en Python, ofreciendo una solución efectiva dentro de los recursos y limitaciones que se presentan en la organización.

This article presents a detailed investigation into the optimal encryption method to safeguard sensitive information in the database of ”Casa Monarca - Ayuda humanitaria al migrante A.B.P.” a shelter in Monterrey for migrants. The AES standard is selected due to its computational efficiency and robust security, and the AES-GCM cipher mode is chosen for its ability to provide confidentiality and authenticity. Implementation involved a relational cloud database and a user interface developed with HTML, CSS, and JavaScript, with the backend implemented in Python, offering an effective solution within the organization’s resources and constraints.

2. Introducción

La circulación masiva de datos ha incrementado en organizaciones públicas y privadas en los últimos años, ha generado diversas discusiones sobre la importancia de implementaciones diversas para salvaguardar los datos personales de las personas involucradas. Una técnica muy utilizada son los sistemas de encriptación modernos. Aunque, la encriptación de datos ha estado presente desde tiempos remotos tal como el sistema de cifrado de Julio César o el de transposición de los espartanosEnc (2023)-, la llegada de nuevas tecnologías ha generado la necesidad de buscar nuevos algoritmos más robustos y sostenibles.

Una problemática reciente que ha emergido debido al constante almacenamiento de información personal, es la exposición frágil de los datos de ciudadanos inmigrantes y las consecuencias que esto puede conllevar. Un ejemplo básico es que, una persona que logre interceptar una información confidencial (dirección de país de origen, nombre, datos financieros, historial médico, entre otros) del afectado, puede llegar a dañar no solo su integridad individual sino también de sus seres cercanos.

Dicho lo anterior, el objetivo de este trabajo se verá enfocado principalmente en proponer un sistema de protección de datos de los inmigrantes que llegan a visitar el albergue *asa Monarca - Ayuda humanitaria al migrante A.B.P.*, en el cual se priorizará en implementar las siguientes iniciativas de seguridad con los datos trabajos: confidencialidad, integridad, autenticación y no repudio IBM (2024). Más adelante se abordarán a detalle la metodología que se utilizó para la elaboración del trabajo.

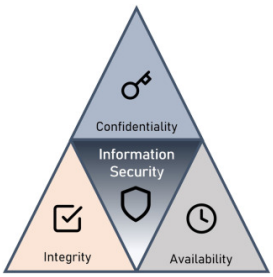

Figure 1: Estándar general de la seguridad de la información en algoritmos criptográficos modernos

3. Marco Teórico

3.1. Esquemas de protección de datos modernos

Antes de proponer estándares o algoritmos criptográficos para cifrar los datos tratados, es esencial ejecutar un *framework* o estructura de la seguridad de la información. Los *Frameworks* de Ciberseguridad son conjuntos predefinidos de políticas, protocolos y requisitos definidos para mejorar estrategias de seguridad y poder proteger de vulnerabilidades o amenazas existentes cib (2019). Entre los más aclamados por la industria se encuentran los siguientes:

- **Triada CIA**: Un modelo de ciberseguridad que permite proteger a los datos con base en tres principios Fruhlinger (2020):
  1. **Confidencialidad**: Solo los usuarios y procesos autorizados deberían poder acceder o modificar los datos.
  2. **Integridad**: Los datos deben mantenerse en un estado correcto y nadie debe poder modificarlos indebidamente, ya sea de forma accidental o maliciosa.
  3. **Disponibilidad**: Los usuarios autorizados deberían poder acceder a los datos siempre que lo necesiten.
- **NIST** *(National Institute of Standards and Technology)*: El Framework NIST ofrece directrices para mejorar la ciberseguridad en organizaciones gubernamentales y privadas. Se centra en *identificar*, *proteger*, *detectar*, *responder* y *recuperarse* de amenazas cib (2019).
- **PCI DSS** *(Payment Card Industry Data Security Standard)*: Es un modelo muy utilizado para organizaciones trabajando con datos transaccionales o de pago. Se encarga de establecer requisitos y protocolos cruciales para la protección de tarjetas de crédito y débito, lo cual ayuda a mitigar los robos de informaciónPCI (2023).

Cada uno de los modelos anteriores ofrecen ventajas y desventajas al cliente, lo cual dependerá principalmente de sus necesidades y prioridades. Para el uso que se le dará en este trabajo, se seleccionará la estructura de la *Triada CIA*, y se explicará con más detalle en el apartado del **??**.

3.2. *Estándares de encriptación modernos*

Estándares y algoritmos de encriptación modernos:

- **Triple DES**: El algoritmo Triple DES (Data Encryption Standard) es un algoritmo de cifrado que se encarga de aplicar el proceso de cifrado tres veces consecutivas usando tres claves diferentes. En el pasado fue muy utilizado, especialmente en los sistemas de pago y transacciones financieras, cajeros automáticos, implementaciones de VPN, sistemas de firewalls (para cifrar las comunicaciones entre dispositivos en la red) y, lo más importante durante este proyecto, protección de datos sensibles en entornos corporativos y gubernamentales.

  El Triple DES funciona en tres pasos: Encriptar-Decencriptar-Encriptar (EDE). Se realiza tomando tres claves de 56 bits (K1, K2 y K3) conocidas como un paquete de claves y encriptando primero con K1, luego desencriptando con K2 y finalmente encriptando una última vez con K3 (Figura 2). La longitud total de la clave asciende a 168 bits, pero según la mayoría de los expertos, su fuerza de clave efectiva es de solo 112 bits. Cobb (2023)

  Un ataque al que son propensos son los llamados *meet-in-the-middle*, consisten en cifrar desde un extremo, descifrar desde el otro y buscar coincidencias entre las claves que producen la misma salida. Aunque se use Doble DES o cualquier otro cifrado repetido, el cifrado doble solo sería tan fuerte como el mismo cifrado ejecutado una vez, pero con una clave que fuera un bit más larga.

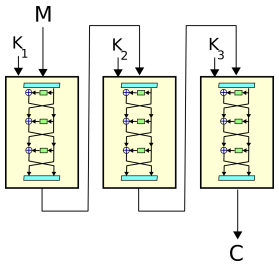

Figure 2: Triple DES

- **AES**: es un algoritmo de cifrado ampliamente utilizado para proteger datos confidenciales. Fue seleccionado por el Instituto Nacional de Normas y Tecnología (NIST) de Estados Unidos como estándar para cifrado simétrico en 2001, reemplazando al ya obsoleto algoritmo DES (Data Encryption Standard) mencionado anteriormente.

  AES opera mediante la realización de una serie de transformaciones matemáticas en bloques de datos de 128 bits, utilizando claves de cifrado de 128, 192 o 256 bits. Cada bloque de datos se considera como una cuadrícula de 16 bytes (4 bytes x 4 bytes) dispuestos en columna. El proceso de cifrado consta de varios pasos, incluyendo SubBytes, ShiftRows, MixColumns y Add Round Key, que se repiten en múltiples rondas dependiendo de la longitud de la clave.
  - El paso SubBytes realiza una sustitución utilizando una tabla de búsqueda conocida como S-box, donde cada byte se sustituye por otro byte de acuerdo con ciertas reglas.
  - El paso ShiftRows desplaza las filas de la cuadrícula en un número específico de veces.
  - MixColumns realiza una multiplicación de matriz para cambiar la posición de cada byte en la columna.
  - Add Round Key hace una operación XOR entre el resultado de los pasos anteriores y una clave de ronda correspondiente.

  El proceso de descifrado sigue pasos similares pero en orden inverso, utilizando operaciones como Inverse MixColumns e Inverse SubBytes para deshacer las transformaciones realizadas durante el cifrado. AES se utiliza en una amplia gama de aplicaciones, desde la seguridad de redes inalámbricas y comunicaciones en línea hasta el cifrado de datos almacenados en dispositivos y servicios en la nube.

  Este algoritmo se considera resistente a todos los ataques, excepto a la fuerza bruta.

  Gee (2023)

- **RSA**: Es un algoritmo de cifrado de clave pública (asimétrico) popular. Utiliza un par de claves: la clave pública, que se utiliza para cifrar el mensaje antes de enviarlo, y la clave privada, que se utiliza para descifrar el mensaje una vez que ha sido recibido (Figura 3).

  La seguridad del algoritmo RSA se basa en la dificultad de factorizar el producto de dos números primos grandes, lo que hace que sea computacionalmente difícil para un atacante determinar la clave privada a partir de la clave pública.

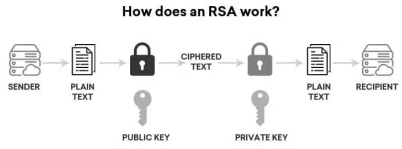

Figure 3: Funcionamiento del algoritmo RSA

- **Blowfish**: Un cifrado simétrico que opera en bloques de 64 bits y usa claves de tamaño cambiante, el proceso que sigue es:

  1. **Expansión de clave**: Comienza con una clave secreta que puede tener una longitud de 32 a 448 bits. Luego, la clave de cifrado se prepara y se extiende utilizando la precomputación del array P y las S-boxes para generar una serie de subclaves.
  2. **Generación de subclaves**: La clave extendida se divide en partes más pequeñas - los bloques de 64 bits se dividen en dos fragmentos de 32 bits. Estos se mezclan con algunos valores predefinidos para crear un nuevo conjunto de subclaves.

3. **Encripción de datos**: Esas dos mitades de 32 bits pasan por 16 rondas de cifrado. Cada ronda implica una secuencia compleja de sustituciones y transposiciones (operaciones XOR, adiciones y búsquedas en las S-boxes).
4. **Después de procesarlos:** Después de las 16 rondas de cifrado, las partes revueltas de 32 bits son unidas de nuevo para crear bloques de texto de 64 bits.

La fortaleza de este cifrado radica en su diseño cuidadoso, capacidad para resistir diversos ataques cirptográficos conocidos y su simplicidad y eficiencia lo hacen adecuado para aplicaciones en donde se necesita un alto rendimiento y una seguridad sólida. Blowfish sentó las bases para algoritmos derivados como Twofish, sin embargo, fue reemplazado por algoritmos como AES (por su ausencia de un proceso de estandarización ampliamente aceptado).

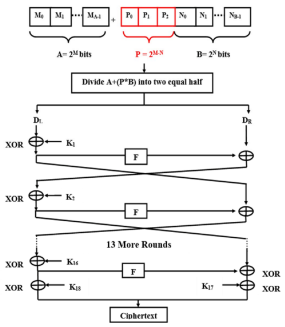

Figure 4: Algoritmo Blowfish

- **Twofish**: Twofish opera en bloques de datos de 128 bits y admite claves de cifrado de longitud variable, desde 128 hasta 256 bits. Utiliza una estructura de red de Feistel similar a la de Blowfish, pero con una mayor cantidad de rondas y operaciones más complejas.

  En cada ronda del algoritmo Twofish, dos palabras de 32 bits desempeñan un papel crucial en la función F. Cada una de estas palabras se descompone en 4 bytes, que luego atraviesan cuatro cajas S dependientes de la clave, caracterizadas por su E/S de 8 bits. La matriz MDS se encarga de combinar los 4 bytes de salida en una palabra de 32 bits. Posteriormente, las dos palabras de 32 bits se combinan mediante un transformador de Hadamard parcial (PHT). En el paso siguiente, estas palabras se suman a dos subclaves de ronda y se realizan operaciones XOR en la mitad derecha del bloque de datos en proceso. Este proceso se repite a lo largo de las rondas del algoritmo, contribuyendo a la robustez y seguridad del cifrado Twofish. En la figura 5 se muestra el diagrama del proceso que sigue.

  Twofish es un algoritmo de cifrado sólido y eficiente que ofrece una seguridad robusta y un rendimiento rápido. Una de las ventajas más grandes es la flexibilidad en la longitud de la clave, lo que lo hace adaptable a diferentes niveles de seguridad, y su implementación eficiente lo hace

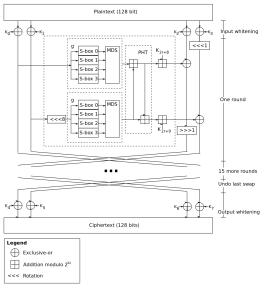

Figure 5: Funcionamiento del algoritmo Twofish

adecuado para una variedad de dispositivos y plataformas. sin embargo, su falta de estandarización y su complejidad podrían limitar su adopción en comparación con otros algoritmos de cifrado más ampliamente reconocidos.

- **Elliptic Curve Cryptography (ECC)**: El algoritmo utilizado como parte del protocolo SSL/TLS cifra la comunicación entre los sitios web y sus visitantes. Proporciona mayor seguridad con claves de menor longitud; una clave ECC de 256 bits proporciona el mismo nivel de seguridad que una clave RSA de 3072 bits.

  Este es un enfoque de cifrado asimétrico que utiliza propiedades matemáticas de las curvas elípticas sobre cuerpos finitos para cifrar y firmar datos de manera segura. A diferencia de otros algoritmos criptográficos, el EEC utiliza operaciones algebraicas sobre puntos en una curva elíptica para generar claves, cifrado y firma digital. Lo anterior es posible verlo en la Figura 6.

  Su ventaja principal es que ofrece un nivel de seguridad comparable con algoritmos de clave pública tradicionales (como RSA mencionado anteriormente) pero utiliza claves más cortas, lo que lo hace útil para dispositivos con recursos limitados, en donde la eficiencia computacional es un factor crítico.

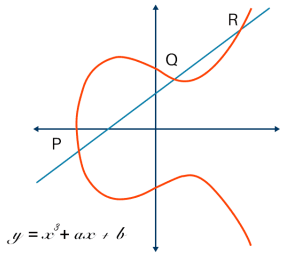

Figure 6: Elliptic Curve Cryptography

3.3. Sistemas de almacenamiento modernos

Una vez que los datos han pasado por el proceso de encriptación es necesario resguardarlos de forma segura, para esto existen diversos dispositivos o servicios de almacenamiento, los cuales priorizan la confidencialidad e integridad de esta sensible información. Entre ellos se encuentran:

- **Dispositivos de almacenamiento físico seguro:** Estos son dispositivos de almacenamiento que suelen ser utilizados para expandir la capacidad de la computadora en la que se resguardan los datos encriptados. Algunos de estos son los discos duros y los discos de estado sólido. Dropbpx (2024)

- **Almacenamiento cifrado de extremo a extremo en la nube:** Estos servicios utilizan el algoritmo de cifrado end-to-end, lo que garantiza que nadie puede acceder a los datos, ya que solo tienen acceso a estos aquellas personas que tienen la clave de descifrado. IBM (2022)

- **Cifrado de datos a nivel de archivo o bases de datos:** Existen algunos sistemas de gestión y gobernanza de datos que permiten encriptar solo aquellos datos meramente sensibles, es decir de algunos archivos específicos, ejemplo de esto es el uso de Transparent Data Encryption (TDE). Lab (2024)

- **Blockchain y DLT:** Estos sistemas almacenan los datos de forma descentralizada, lo que garantiza la seguridad e integridad de estos, es por eso que estos sistemas son utilizados en casos donde es necesario un alto grado de seguridad y transparencia. Acuña (2024)

- **Almacenamiento en dispositivos de IoT:** Estos dispositivos hacen uso del hardware dedicado, además del cifrado de los propios dispositivos IoT en los que están contenidos los datos, esto con la finalidad de aportar mayor confidencialidad. Martínez (2021)

3.4. Bibliotecas disponibles

Para la aplicación de métodos criptográficos se podría usar:

- PyCryptodome: Esta es una biblioteca de cifrado para Python que proporciona implementaciones de varios algoritmos criptográficos, incluidos Triple DES, AES, RSA, Blowfish, Twofish y ECC.

- cryptography: Otra biblioteca popular de Python que ofrece una amplia gama de funcionalidades criptográficas, incluidas implementaciones de AES, RSA y ECC.

Para el desarrollo de la interfaz hay varias opciones disponibles, como:

- Tkinter: Es un framework completamente gratuito para poder desarrollar interfaces, se puede crear un inicio de sesión, poner mensajes en pantalla y poder poner los datos rellenar.

- PyQt: Es igual de útil para el desarrollo de interfaces, pero tiene un costo de licencia, su ventaja es que se pueden desplegar visualizaciones, widgets personalizados y ofrece una conexión sencilla a bases de datos.

4. Metodología

A pesar de haber tenido la opción de utilizar frameworks disponibles para desarrollar la interfaz desde Python, se optó por crear una página web utilizando HTML, CSS y Javascript, junto con el framework Svelte para facilitar la conexión al Back-End que fue realizada con Python y al servidor que se usa para manejar los permisos por usuario, que fue realizado con Javascript.

4.1. Base de datos

Para almacenar la información generada, se optó por utilizar PostgreSQL, un sistema de base de datos de código abierto que funciona a base de SQL, el proyecto es gratuito, contribuyendo así a minimizar los costos. Se escogió este software ya que es capaz de gestionar grandes volúmenes de datos sin complicaciones y su manera de almacenar datos permite conservar su integridad y consistencia. Además, ofrece un Back-End escalable para diferentes tipos de aplicaciones.

Con el fin de escalar el despliegue de la interfaz, se necesita tener una conexión a una base de datos en la nube, esto es beneficioso ya que se permite el acceso a los datos desde cualquier lugar y en cualquier momento, además que se reduce el costo de inversión ya que los proveedores se encargan del mantenimiento del software y hardware, además que ofrecen medidas de seguridad robustas para proteger los datos. Se escogió la base de datos de Google, ya que cumple con una amplia gama de estándares y regulaciones internacionales como GDPR, HIPAA, ISO/IEC 27001, SOC 2/3 entre otras, lo que asegura que los datos se manejan de acuerdo con los requisitos más estrictos, además se cuenta con una gran seguridad tanto física como en la red; los centros de datos de Google están protegidos mediantes controles de acceso biométricos, mientras que en la red Google emplea medidas de seguridad como firewalls, protección DDoS, sistemas de monitoreo y alertas para detectar amenazas en tiempo real, finalmente es un servicio de almacenamiento flexible y escalable atendiendo a las necesidades del usuario, lo que es conveniente para este proyecto debido a la gestión de bases de datos que se tiene.

4.1.1. Costos de Almacenamiento en la Nube

Los costos asociados con el almacenamiento en la nube de Google Cloud SQL se detallan en la siguiente tabla:

Concepto	**Costo**
Costo por almacenamiento en SSD (250 GB)	$28.75
Costo de respaldo para espacio adicional (+5% de incremento semanal)	$4.75
Total	$33.5 dólares

Table 1: Desglose de los costos de almacenamiento en Google Cloud SQL

Estos costos sirven como referencia para a futuro poder realizar un presupuesto y establecer los costos del proyecto.

4.2. Back-End

4.2.1. Creación de llave

Esta llave es necesaria por el método de encriptación utilizado, AES, que utiliza una llave para poder encriptar y desencriptar los datos. Para garantizar el correcto funcionamiento y ofrecer una mayor protección, la constraseña inicial escogida deberá de ser segura.

Se utilizó una contraseña generada aleatoriamente con 50 caracteres. Esta contraseña se encripta al conseguir el hash de cada carácter de la contraseña a través de la función "SHA256". Una vez que ya se cuente con este hash, se produce un valor 'sal', el cual es un valor aleatorio de la misma longitud que se concatenará a la contraseña para solidificar la contraseña.

Después se procede a usar el algoritmo "Password-Based Key Derivation Function 2" (PBKDF2), para conseguir el hash de esta nueva contraseña concatenada. Una vez hecho esto, se codifica en base 64 para convertir la cadena de bytes a texto plano. Finalmente, este texto se almacena en un archivo ".env" para que en nuestro código principal no este expuesto como variable.

Para utilizar esta llave, se lee la variable dentro del archivo ".env", que posteriormente se decodifica en base 64 para obtener la cadena de bytes de la llave.

Cabe recalcar la importancia de no perder o eliminar esta llave de acceso. En caso de que esta se pierda o fuese eliminada, los datos no se podrán desencriptar a la hora de llamarlos de la base de datos. La unica manera para desencriptarlos es con esta llave creada.

4.2.2. Función para encriptar

Esta función es de suma importancia para generar el proyecto creado, pues recibe la llave (definida en el .env para mantenerla segura) y el texto plano (en este caso las respuestas a las preguntas en formato JSON), encripta la información y se manda a la base de datos llamada "migrantes". Aunque la llave nunca cambia, para facilitar el proceso de desencriptación, si se genera un vector de inicialización (cadena random de 16 bytes) cada vez que se corre el código. De tal manera, no presenta un riesgo de cibserseguridad la posible filtración al público (pues no es suficiente para desencriptar la información) pero sí se asegura un encriptado diferente cada vez, lo que es una ventaja. Después el texto plano lo codificamos en "UTF-8".

Después, se creo un Padding de PKCS7 de 128 bits para rellenar el bloque de texto a 128 bits. De esta manera si el texto es muy corto, se rellena con este padding y se puede encriptar. El siguiente paso fue generar el cifrado AES en su modo blockchain, mandandole la llave y el vector de iniciación. Una vez teniendo el cifrado, se aplica al texto con el objetivo de encriptarlo. Finalmente, se devuelve el vector de iniciación y el texto cifrado. El peseudocódigo mostrada en el bloque de *Algorithm 1* muestra la lógica del proceso anterior.

Esta función se llama con un API desde el front-end usando un POST, para poder recibir los datos del formulario, crear recursos, actualizar los recursos e incluso enviar los datos ya creados.

Algorithm 1 AES Encryption Function

```
1:  procedure ENCODING(key, plaintext, lower=True)
2:      iv ← os.urandom(16)                                  ▷ Generate IV
3:      plaintext_formatted ← plaintext.encode("utf − 8")
4:      if lower then
5:          plaintext_formatted ← plaintext_formatted.lower()
6:      end if
7:      padder ← padding.PKCS7(128).padder()
8:      padded_plaintext ← padder.update(plaintext_formatted) +
    padder.finalize()
9:      cipher ← Cipher(algorithms.AES(key), modes.CBC(iv)
10:     encryptor ← cipher.encryptor()
11:     ciphertext ← encryptor.update(padded_plaintext) +
    encryptor.finalize()
12:     return iv + ciphertext
13: end procedure
```

4.2.3. Función para desencriptar

Esta función es similar a la función de encriptación, sin embargo, varía en unas pequeñas cosas. Primero se le manda la llave y el texto cifrado y, teniendo esto, separa el vector de inicialización del texto cifrado. Posteriormente se desencriptan los datos con la creación del mismo cifrado utilizado en la función de encriptación. Por último, cómo se le agregó un padding en la función de encriptación, se eliminó este componente de los datos y se convirtió a texto plano para la correcta visualización de los datos originales dentro de la página web.

Esta función también se llama con un API desde el front-end usando un POST, lo que permite accesar a la base de datos usando SQL, utilizar el código para desencriptar la información y después, con el POST, mandarla de nuevo al HTML para su correcta visualización. El siguiente pseudocódigo muestra la lógica explicada anteriormente.

Algorithm 2 AES Decryption Function

```
1:  procedure DECODING(key, ciphertext)
2:      iv ← ciphertext[: 16]          ▷ Extract IV from ciphertext
3:      ciphertext ← ciphertext[16 :]  ▷ Remove IV from
    ciphertext
4:      cipher ← Cipher(algorithms.AES(key), modes.CBC(iv), backend
5:      decryptor ← cipher.decryptor()
6:      padded_plaintext ← decryptor.update(ciphertext) +
    decryptor.finalize()                          ▷ Decrypt ciphertext
7:      unpadder ← padding.PKCS7(128).unpadder()
8:      plaintext ← unpadder.update(padded_plaintext) +
    unpadder.finalize()                           ▷ Remove padding
9:      return plaintext
10: end procedure
```

4.3. Función para generar ID

La función para generar un ID es muy simple, primero se define un conjunto de caracteres que incluye letras (mayúsculas y minúsculas) y dígitos. Después, se genera una cadena aleatoria de seis de estos caracteres, misma que se encripta utilizando

la función de encriptación mencionada anteriormente. Finalmente, se devuelve el ID encriptado, listo para ser usado como un identificador único y seguro en la base de datos.

4.3.1. Función para subir datos a la DB

Esta función es llamada dentro de la función de encriptación y su trabajo es esencial, pues es la que inserta en la base de datos la información encriptada. Primero, se definen las columnas existentes en la tabla, después se crea la columna de ID que usa la función mencionada previamente para agregar un identificador a cada registro. Después, se encriptan los valores usando la función anterior y, finalmente, se construye una consulta SQL INSERT en donde se insertan los valores encriptados a la base de datos creada. Finalmente, se cierra el cursos y la conexión con la base de datos.

4.3.2. Función para extraer datos de DB

Se encarga de obtener y desencriptar los datos de un usuario específico desde una tabla en la base de datos. Primero, se establece una conexión a esta para después hacer una consulta usando SQL para obtener todos los datos de la tabla especificada, y guardar la información en un DataFrame de Pandas.

Teniendo esto, la función se encarga de recorrer cada fila del DataFrame y desencripta cada valor que se encuentra usando la función de desencriptación. Si el valor es un objeto memoryview, se convierte a bytes antes de la desencriptación para asegurar su correcto funcionamiento. Estos valores desencriptados se almacenan en un diccionario nuevo para concatenarlo con las filas desencriptadas. Finalmente, se cierra la conexión a la base de datos, devolviendo así el DataFrame con los datos desencriptados.

4.3.3. Función buscar migrante

La función recibe el nombre completo y fecha de nacimiento en formato JSON y procede a validar los datos, revisando que no falte ninguno. Después, se desencriptan los datos almacenados en la tabla de migrantes y almacena los datos desencriptados en un nuevo DataFrame. La función continúa buscando al migrante usando una función auxiliar $find\_most\_similar\_name$ que calcula la distancia de Levenshtein entre el nombre ingresado y todos los nombres en la base de datos, con el objetivo de encontrar el registro del migrante aunque tenga faltas de ortografía o errores al escribirlo. Esta función, también conocida como "distancia de edición", mide la diferencia entre dos cadenas de caracteres calculando el número mínimo de operaciones necesarias para transformar una cadena en otra. Las operaciones permitidas son inserción, eliminación y sustitución de un solo carácter. El algoritmo funciona creando una matriz donde las filas y columnas representan los caracteres de las dos cadenas que se están comparando. Así, la primera fila y la primera columna de la matriz se inicializan con los índices de sus posiciones respectivas, lo que representa el costo de convertir una cadena vacía en la otra mediante inserciones o eliminaciones.

Luego, se recorren las dos cadenas carácter por carácter y se calcula el costo de sustitución para cada par de caracteres: es 0 si los caracteres son iguales y 1 si son diferentes. Cada celda en la matriz se rellena tomando el mínimo entre tres valores: el valor de la celda superior más 1 (costo de una eliminación), el valor de la celda izquierda más 1 (costo de una inserción) y el valor de la celda diagonal más el costo de sustitución. Al finalizar, el valor en la celda inferior derecha de la matriz contiene la distancia de Levenshtein entre las dos cadenas. Por ejemplo, para transformar "gato" en "pato", la distancia de Levenshtein es 1, ya que solo se necesita una sustitución (reemplazar 'g' por 'p').

Luego, filtra el DataFrame para encontrar el registro del migrante que coincida con la fecha de nacimiento y el nombre más similar. Si se encuentra el migrante, la función devuelve una respuesta con los datos del migrante en formato JSON. Si no se encuentra el migrante, se devuelve un error indicando que el usuario no fue encontrado.

4.3.4. Función editar estatus del migrante

Si una persona salió de Casa Monarca, es importante mantener un registro para saber quienes son los miembros activos e inactivos dentro del establecimiento. Para esto, se creó una función a la que solo tienen acceso los administradores que se encarga de recibir una solicitud para editar los datos de un migrante específico, validando y desencriptando la información almacenada en la base de datos. En primer lugar, la función obtiene los datos enviados (nombre completo, fecha de nacimiento y último día en Casa Monarca) en formato JSON a través de request.json.

A continuación, la función desencripta los datos almacenados en la tabla migrantes utilizando funciones mencionadas anteriormente y almacena los datos desencriptados en un DataFrame. También se utiliza la función de similitud para encontrar el nombre más similar al proporcionado y se filtra el DataFrame para encontrar el registro del migrante que coincida con la fecha de nacimiento y el nombre más similar. Si se encuentra el migrante, la función llama a otra función que se encarga de actualizar los datos. Primero, encripta cada campo de los datos y verifica que los campos $activo\_CM$ y ultimaFecha existen en los datos encriptados. Luego, establece una conexión a la base de datos y obtiene el ID encriptado del migrante. Finalmente, ejecuta una consulta SQL para actualizar los campos $activo\_CM$ (de tener un valor de si obtiene ahora un valor de no) y ultimaFecha (de estar vacío ahora tiene la última fecha en Casa Monarca) en la tabla migrantes. Al finalizar, se cierra la conexión a la base de datos.

4.3.5. Función para conteo de ingresos

La función se encarga de contar y categorizar las entradas de migrantes para una fecha específica, clasificándolos por tipo (Adulto, niño, etc.) y sexo (Hombre, mujer, etc.). Primero, obtiene y desencripta la información de la tabla migrantes y filtra a los registros correspondientes a la fecha que ingresó el usuario. Luego, los tipos de migrantes se normalizan, si aparece "niño no acompañado" o "niña no acompañada" se agrupa como niño o niña respectivamente y se crea una categoría relacionada a la comunidad LGBTQI+. Al final, se cuentan los migrantes

clasificados como niños, niñas, hombres, mujeres, comunidad LGBTTIQ+, y se calcula el total al sumar estas categorías. Finalmente, se crea un DataFrame que se encripta y se envía a la base de datos "ingresos", Cómo solamente se envían los campos nuevos, no hay un problema de sobreescritura y todo funciona de una manera correcta.

4.3.6. Modelo predictivo

Cómo se están manejando fechas y cantidades, quiere decir que se está trabajando con series de tiempo. Para su predicción, se utilizó el modelo Prophet de Facebook, un modelo que mezcla SARIMA e inteligencia artificial para detectar tendencia, estacionalidad y días festivos y poder generar predicciones acertadas Alonso-Cortés and Arribas (2022). Es un modelo aditivo que tiene la siguiente fórmula en función del tiempo:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Donde $y(t)$ es la predicción, $g(t)$ es el componente tendencia, $s(t)$ es la estacionalidad, $h(t)$ es el componente de eventos que impactan la serie (vacaciones, días festivos, asuetos) y ϵ_t es el error irreducible.

Ahora, la implementación es sencilla, se le manda la fecha junto con el conteo correspondiente de ingresos para cada tipo de población y los días a predecir y se crea el modelo predictivo.

4.4. Desarrollo de Tablero

Con tanta información disponible, se puede realizar un tablero con información descriptiva sobre la información de los migrantes.

Para poder interpretar y utilizar los datos dentro de un tablero, se importan los datos usando un script de Python, en el cual se desencriptan los datos del servidor en PostgreSQL y se procede a realizar la lectura de estos, se utilizan las mismas funciones ya mencionadas, pero ahora se corren dentro de PowerBI.

Se procedió a crear una nueva tabla para poder tener un conteo de cuántos registros (migrantes en este caso) provienen de cierto país.

El tablero cuenta con 7 indicadores:

- Conteo de migrantes activos
- Conteo de migrantes inactivos
- Mapa relleno con los paises de origen de los migrantes
- Desglose del tipo de población: Adulto(a), Niño(a), etc
- Servicios más populares: Gráfico de barras visualizando los servicios más populares entre los migrantes.
- Conteo de migrantes por edad:

Esta información se puede presentar gracias al acceso a la base de datos desencriptada.

4.5. Front-End

La interfaz se realizó usando HTML, CSS y Javascript, junto con el framework Svelte para facilitar la conexión entre el back-end y front-end así como hacer más sencillo el proceso de hosting la página web. Tiene un login de usuario y contraseña, la opción de registrar nuevos usuarios (pidiendo una contraseña del administrador), y dependiendo de los permisos que se tienen son las acciones que se pueden tomar.

4.5.1. Jerarquías y permisos

Se tienen tres niveles de usuario.

- **Usuario (Nivel 1):** Este tipo de usuario solo tiene acceso a dar de alta un nuevo registro y buscar un registro.
- **Servicios (Nivel 2):** Los usuarios que cuentan con este nivel de permisos tienen la posibilidad de buscar entre los registros, dar de alta un nuevo ingreso, así como editar un registro para modificar los servicios que ha recibido la persona.
- **Administrador (Nivel 3):** El administrador es el único que tiene permitido crear usuarios así como modificar sus permisos y marcar a migrantes como inactivos en Casa Monarca. Además, es el único que tiene acceso a las estadísticas sobre el flujo y estatus de cada persona que se encuentra en el refugio. Finalmente cuenta con el resto de los permisos como lo es dar de alta un nuevo registro, modificar el estatus de un migrante en caso de que salga del refugio, buscar algún registro y editar los datos de un migrante al recibir nuevos servicios.

Figure 7: Menú Usuario (Nivel 1)

4.5.2. Inicio de sesión

Se cuenta con un inicio de sesión con un login y contraseña. Se verifican los permisos a través de un script de javascript que maneja la autenticación de usuarios utilizando Firebase Authentication y Firestore. Define "writable stores" para mantener el estado de autenticación y la información del usuario, y proporciona el manejo para registrar nuevos usuarios, iniciar sesión y cerrar sesión. También esta al tanto cambios en el estado de autenticación para actualizar las "stores" correspondientes con la información del usuario autenticado, incluyendo la asignación de un rol predeterminado ("usuario") y la gestión de sus datos en Firestore.

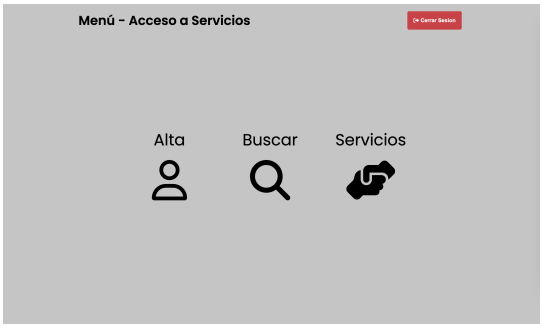

Figure 8: Menú Servicios (Nivel 2)

Figure 9: Menú Administrador (Nivel 3)

El HTML y CSS solo se encargan de proporcionar una interfaz simple y fácil de usar para el usuario, permitiendo así a los usuarios el registrarse e iniciar sesión. Esa página en específico usa ”writable stores” para mantener el estado del usuario y sus datos, y maneja las acciones de autenticación, incluyendo la verificación de contraseñas y la asignación de roles. En general, la interfaz cambia dinámicamente entre los modos de registro e inicio de sesión, muestra mensajes de error cuando es necesario y utiliza estilos CSS para mejorar la experiencia del usuario, para que el diseño se vuelva amigable y ameno para los usuarios.

4.5.3. Creación de usuarios

El registro para un nuevo usuario está dentro de la página principal, cualquier persona puede crear un nuevo usuario poniendo su correo electrónico, contraseña y, lo más importante de todo que se hizo para mantener un nivel de seguridad en la información y plataforma creada, una contraseña del administrador que solo tendrá una persona de Casa Monarca a nivel local y escrita en papel, que será el encargado de dar el permiso para la creación del nuevo usuario. El nuevo usuario tiene un permiso de nivel 1 (el más bajo) y se necesita a un administrador para otorgar permisos más altos y tener más accesos en la página.

4.5.4. Nuevo migrante

El primer botón dentro de la página de navegación se encarga de implementar un formulario detallado para la entrevista de ingreso al albergue Casa Monarca (misma que tenía el Socio Formador anteriormente en Kobo) con el objetivo de manejar la captura y procesamiento de datos del usuario. Permite a los usuarios ingresar diversos datos personales y de contacto, así como información relevante sobre su situación migratoria. Las primeras preguntas son obligatorias y, al final, los datos capturados se almacenan en un objeto answers que se manda como JSON y se encriptan en el servidor de Python que cuenta con las funciones anteriormente mencionadas mediante una solicitud POST. La interfaz también incluye manejadores para subir fotos, manejar cambios en los checkboxes y radio buttons, y gestionar mensajes de éxito o error según el resultado de la solicitud. Además, se ofrece la posibilidad de navegar de regreso al menú de administración o cerrar sesión.

Las fotos que son ingresadas se codifican en base64 antes de ser almacenadas y enviadas a Python, en donde se encripta la codificación obtenida.

4.5.5. Búsqueda de migrante

Esta página sirve para desplegar la información de un migrante específico. Para esto, primero pregunta por nombre y fecha de nacimiento (que se procesan usando las funciones de Python mencionadas en 4.2.2), y se encarga de mostrar la información que le corresponde según su ID.

Esta página muestra las 3 fotos (en donde se decodifica en base64 y se llama imprimiendo su información), así como los datos dentro de las dos bases de datos, migrantes y servicios. Entonces, del lado izquierdo se imprime la información de la entrevista inicial (base de datos ”migrantes”) y del lado derecho aparecen los servicios que se le han otorgado a la persona (base de datos ”servicios”). Únicamente si el usuario es un administrador, se desplegará también quién dio dicho servicio y cuando lo dieron, las demás jerarquías (usuario y servicios) solo podrán ver los servicios que se han otorgado al migrante.

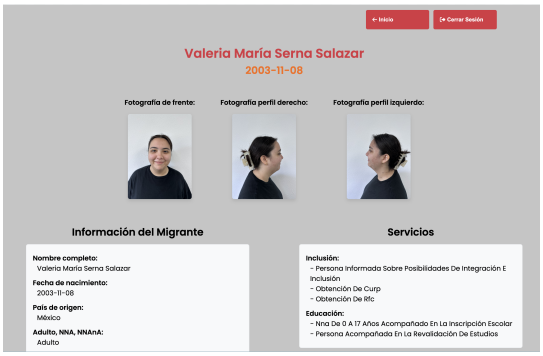

Figure 10: Buscar sin ser Administrador

4.5.6. Agregar a servicios

Al agregar servicios existen 5 departamentos a los que se pueden ingresar: inclusión, reubicación, educación, empleabilidad y salud mental. Cada menú despliega diferentes opciones en formato de ”checkbox” para facilitar el llenado de los datos y cada respuesta va a la base de datos de servicios que está conectada con el ID, nombre y otros datos de la base de datos de migrantes. Así, se mantiene la conexión entre ellas y se asegura que se trata de la misma persona en ambas bases de datos.

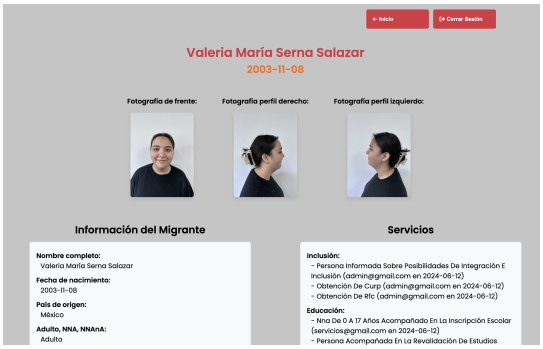

Figure 11: Buscar siendo Administrador

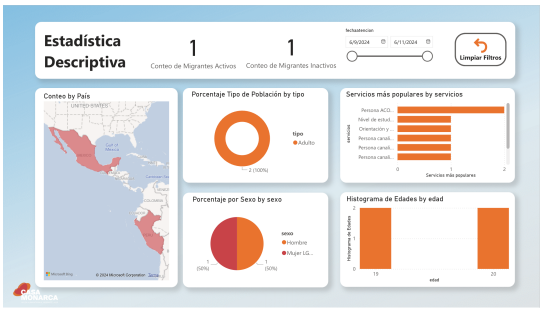

Figure 12: Dashboard

Al dar enviar a los servicios brindados se manda al código de Python que encripta la información y luego se sube a la base de datos.

4.5.7. Cambiar permisos de usuario

Este botón solo aparece para los administradores y al darle click se despliega un ”drop down menu” que muestra todos los usuarios que están dados de alta en la página con su permiso actual entre paréntesis, abajo se despliegan las opciones del nuevo permiso que se busca dar, al dar enviar se actualiza la base de datos y se procede a la asignación de su nueva jerarquía y permisos. Puede ser que de un usuario nivel 1 pase a ser nivel 3 o viceversa, también se pueden disminuir permisos de nivel 3 a nivel 1, por ejemplo.

4.5.8. Cambiar estatus de migrante

Considerando que se trata con una parte de la población vulnerable y, aunque los migrantes salgan de Casa Monarca se debe de mantener su registro en la base de datos, al ingresar a esta página se pide el nombre completo y fecha de nacimiento. Cuando encuentra al migrante pregunta por la última fecha en Casa Monarca y, en la base de datos de ”migrantes” actualiza 2 columnas, una llamada ”activo_CM” que toma un valor binario (si o no) y aunque está en default a si, cuando se envía el cuestionario con la fecha de salida esta cambia a no y tambien se guarda la fecha de salida en otra columna.

4.5.9. Tablero con estadística descriptiva

Se creó un dashboard usando PowerBI en donde se muestra la información más importante de los datos recabados durante el proceso. Para lograrlo, se corre el script de python que desencripta la información dentro de PowerBI, obtneiendo así los datos desencriptados durante el desarrollo del tablero. El resultado se muestra en la figura 12. Cabe aclarar que solo los administradores tienen acceso a estos datos.

4.5.10. Predicción de migrantes

En esta sección de la interfaz, se cuenta con dos botones, el primer botón es para realizar el conteo de los migrantes. Desencripta la base de datos de ”ingresos” y verifica que fecha tiene faltantes. Posteriormente, desencripta la base de datos de ”migrantes” y utiliza la función de 4.3.5 para realizar el conteo y subirlo a la base de datos. El segundo botón conssite de un ”slider” donde se selecciona la cantidad de días a predecir este dato se manda al back-end y corre el modelo predictivo. El resultado es una gráfica con los ingresos estimados.

5. Resultados

Recordemos que el alcance de este proyecto, es salvaguardar todos los datos de los registros de migrantes por medio de la institución casa Monarca. Es por eso, que presentaremos un poco los resultados obtenidos.

5.1. Datos encriptados en servidor en la nube

Para mejorar la escalabilidad y accesibilidad de nuestra aplicación, decidimos migrar nuestra base de datos local de PostgreSQL al servidor Google Cloud SQL. A continuación se detallan los pasos realizados durante el proceso de migración:

1. Creación de una instancia en Google Cloud SQL: Inicialmente, configuramos una nueva instancia de PostgreSQL en Google Cloud SQL, seleccionando la configuración adecuada de CPU y almacenamiento según nuestras necesidades.
2. Exportación de la base de datos local: Utilizamos el comando `pg_dump` para exportar todas las tablas de nuestra base de datos local.
3. Importación de datos en Google Cloud SQL: Subimos el archivo de copia de seguridad a un bucket de Google Cloud Storage y, posteriormente, importamos los datos a nuestra nueva instancia de Cloud SQL utilizando el comando de importación de SQL.
4. Verificación y pruebas: Tras la migración, realizamos diversas pruebas para asegurar que todos los datos se habían transferido correctamente y que la base de datos funcionaba como se esperaba en su nuevo entorno en la nube.

En la figura13,se muestra el tablero gráfico de la base de datos creada dentro de Google Cloud Service.

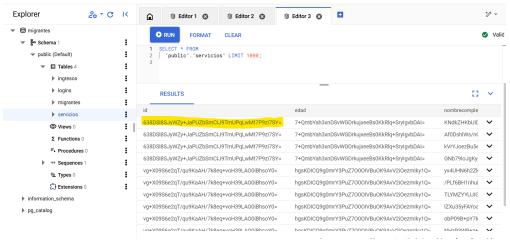

Figure 13: Ejemplo de datos encriptados dentro de la tabla *servicios*

5.2. *Encriptación*

Como se mencionó antes, no se requiere ningún método de encriptación de un tercero, ya que al enviar los datos en plano del formulario, automáticamente nuestra función las encripta. En la tabla 3, se muestra un como se vería la encriptación de un texto en plano.

Texto plano	Texto cifrado
Rodrigo Perez Perez Gonzalez	”YrhsTxM5yOWwQUgvr9O baiPxVtbCxD/SWK0nTK68 P8cowxpoSgZeXS5hIvTQlX5m”

Table 2: $plain_text \rightarrow cipher_text$

5.3. *Cálculo de distancias de Levenshtein*

Una parte fundamental del proyecto es la búsqueda de un nombre que nos permita encontrar resultados aunque tengan faltas de ortografía. Para esto, se utilizó el cálculo de las distancias de Levenshtein cuyo funcionamiento es explicado en 4.3.3. Se corrió el algoritmo en Python con diferentes faltas ortográficas para comparar las distancias y asegurar su correcto funcionamiento. El resultado obtenido, al comparar ”Rodrigo Perez Gonzalez” con las siguientes búsquedas fue:

Búsqueda	Distancia
”Rodrigo Pérez González”	2
”Rodrigo Gonzales Perez”	11
”Rodrigo Perez Gonz”	6
”Rodrigo Perezz Gonzalez”	1

Table 3: Distancias de Levenshtein

Es posible ver que entre menor es su distancia, menor es la diferencia y más probabilidad hay de que esa búsqueda específica haya sido la que se buscaba en un inicio. Cabe aclarar que si el nombre se busca de manera correca, la distancia es 0 y ese es el que se elije.

5.4. *Modelo predictivo*

Se nos otorgaron datos desde el 2 de febrero del 2024 hasta el 24 de febrero del mismo año, se realizó el modelo predictivo para predecir 48 días a futuro para estar alineado con la fecha actual. En la figura 14 se ven los resultados, el modelo replicó la tendencia, puede que sea repetitiva, pero conforme se alimente de datos reales, la precisión mejorará considerablemente.

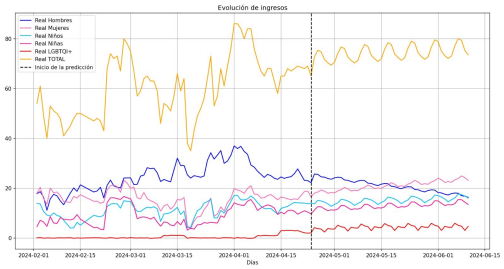

Figure 14: Resultado del modelo.

6. Conclusiones

En este proyecto, se desarrolló una solución integral para proteger la información sensible de los migrantes atendidos por Casa Monarca. La implementación de un sistema robusto de encriptación y una base de datos en la nube garantiza tanto la seguridad como la accesibilidad de los datos.

6.1. *Principales logros*

1. Encriptación eficiente y segura

Se seleccionó el estándar AES con el modo de cifrado AES-GCM, logrando la protección eficaz de los datos mediante una encriptación que asegura confidencialidad y autenticidad. Este enfoque permite que los datos sean encriptados automáticamente al ser ingresados en el sistema, sin necesidad de intervención adicional de terceros.

2. Migración a la nube

La base de datos local de PostgreSQL fue migrada exitosamente a Google Cloud SQL, mejorando la escalabilidad y accesibilidad del sistema. Este proceso incluyó la creación de una nueva instancia en la nube, la exportación e importación de datos, y pruebas exhaustivas para asegurar la integridad y funcionalidad de la base de datos en su nuevo entorno.

3. Desarrollo de un modelo predictivo y dashboard de estadísticas

Se implementó un modelo predictivo capaz de anticipar tendencias a futuro en la llegada de migrantes. El modelo muestra repetitividad, el cual puede ser indicación de un área de mejora en cuanto a la precisión de este. Por otra parte, se implementó un tablero interactivo utilizando PowerBI para visualizar estadísticas descriptivas importantes de los migrantes. Este tablero desencripta la información en tiempo real para proporcionar a los administradores acceso a datos críticos, facilitando la toma de decisiones informadas.

6.2. *Impacto y futuro*

La implementación de este sistema no solo asegura la protección de datos sensibles, sino que también mejora la eficiencia operativa de Casa Monarca. La migración a la nube y el uso de tecnologías modernas de encriptación y análisis de datos proporcionan una base sólida para futuras expansiones y mejoras del sistema.

References

Frameworks de ciberseguridad, 2019. URL https://www.cec.es/frameworks-de-ciberseguridad-tipos-estrategias-implementacion-y-beneficios/.

A brief history of encryption (and cryptography), 2023. URL https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption.

Advanced encryption standard (aes), May 2023. URL https://www.geeksforgeeks.org/advanced-encryption-standard-aes/.

Payment card industry (pci) data security standard (dss), 2023. URL https://learn.microsoft.com/en-us/compliance/regulatory/offering-pci-dss.

Wilder Pereyra Acuña. ¿qué es blockchain o tecnología de registro distribuido (dlt)?, 2024. URL https://blogposgrado.ucontinental.edu.pe/blockchain-tecnologia-registro-distribuido-dlt.

Marina Alonso-Cortés and Victoria Arribas. Análisis y predicción de series temporales con fb prophet python, April 2022. URL https://www.modeldifferently.com/2022/04/analisis_prediccion_ts_prophet/.

Michael Cobb. What is triple des and why is it being disallowed?: Techtarget, Jan 2023. URL https://www.techtarget.com/searchsecurity/tip/Expert-advice-Encryption-101-Triple-DES-explained.

Dropbpx. Tipos de dispositivos de almacenamiento, 2024. URL https://experience.dropbox.com/es-la/get-organized/storage-devices.

Josh Fruhlinger. The cia triad: Definition, components and examples, 2020. URL https://www.csoonline.com/article/568917/the-cia-triad-definition-components-and-examples.html.

IBM. ¿qué es el cifrado end-to-end?, 2022. URL https://www.ibm.com/mx-es/topics/end-to-end-encryption.

IBM. What is encryption?, 2024. URL https://www.ibm.com/topics/encryption.

Kaspersky Lab. ¿qué es el cifrado de datos?, 2024. URL https://www.kaspersky.es/resource-center/definitions/encryption.

Guillermo Martínez. Encriptación y cifrado de datos en plataformas iot basadas en fiware. *Universidad de Sevilla*, 2021.