Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

T E S I S

Sistema de pase de lista con cifrado híbrido, para la alumnos universitarios

Presenta: Leslie Abril Gómez Mora

Para obtener el grado de: Licenciatura en Ingeniería en Ciencias de la Computación

Director: Dr. Juan Francisco Leyva Bonilla

Codirector: Dr. Héctor David Ramírez Hernández

Puebla, Pue, Noviembre 2025

Dedico este trabajo a mi familia y a todos mis seres amados, por su amor y apoyo incondicional, en especial al panque.

Agradecimientos

Deseo manifestar mi más sincero agradecimiento a mis profesores y seres queridos, quienes me brindaron su apoyo incondicional a lo largo del desarrollo de este proyecto. De manera muy especial, extiendo un sincero reconocimiento al Dr. Juan Francisco Leyva Bonilla, por haber formado parte de esta investigación y por guiarme con paciencia y un profundo amor por su profesión. Su ejemplo ha sido fundamental en la construcción del camino que hoy sigo como especialista en el área. Asimismo, agradezco profundamente al Dr. Héctor David Ramírez Hernández, por confiar en mi capacidad y por aportar su invaluable conocimiento a este trabajo de tesis, enriqueciendo significativamente cada etapa del proceso. Finalmente, mi más sentido agradecimiento a la Mtra. María del Consuelo Molina García, cuya calidez humana y vocación docente hicieron que la facultad se convirtiera, para mí, en un segundo hogar durante toda mi formación profesional.

Índice general

Capítulo 1

Introducción

1.1. Planteamiento del problema

En el contexto universitario, el proceso de pase de lista es una práctica común que los profesores realizan para registrar la asistencia de los estudiantes. Sin embargo, este proceso suele ser manual y consume tiempo valioso de las clases, ya que puede incluir desde la lectura en voz alta de los nombres de los alumnos hasta la firma de listas físicas. Según un estudio realizado por el Instituto Nacional para la Evaluación de la Educación (INEE), el 97.8 % de los profesores en instituciones de educación media superior en México utilizan métodos manuales para tomar asistencia, lo que representa una pérdida de aproximadamente 10 a 15 minutos por clase [Mancera Corcuera, 2016]. Esta metodología tradicional no solo es ineficiente, sino que también limita el tiempo disponible para actividades académicas más productivas, afectando el rendimiento tanto de los profesores como de los estudiantes.

Además, en situaciones de emergencia, como siniestros o evacuaciones, no existe un sistema automatizado que posibilite conocer en tiempo real la ubicación y el número de personas dentro de los edificios. Según un estudio de la Universidad Nacional Autónoma de México (UNAM), entre 1990 y 2016, México presentó un promedio anual de más de 700 sismos con distintas magnitudes, además de que en México ocurren en promedio 1,200 incendios al año en instituciones educativas, y en el 60 % de estos casos, no se cuenta con un sistema eficiente para verificar la evacuación completa de los edificios [Pérez-Gavilán et al., 2018]. Esta falta de información puede dificultar la toma de decisiones rápidas y efectivas, comprometiendo la seguridad de la comunidad universitaria.

Otro problema relevante es la falta de acceso inmediato a información en

caso de incidentes, como datos médicos de los estudiantes (alergias, tipo de sangre, contacto de emergencia, etc.). Un estudio realizado por el Servicio de Urgencias de Hospitales en España destaca cómo la historia clínica informatizada optimiza la seguridad y el acceso a la información, facilitando la gestión clínica y optimizando los tiempos asistenciales en un entorno donde la rapidez y la precisión son cruciales [Busca and Marrón, 2010]. Aunque la digitalización de estos datos podría resolver este problema, también plantea un desafío importante: la protección de información sensible. En México, según el Instituto Nacional de Transparencia, Acceso a la Información y Protección de Datos Personales (INAI), en 2023 hubo un aumento exponencial en las actividades maliciosas como robo de datos o secuestro de información sensible detectadas que experimentaron un crecimiento del 950 % [Sánchez, 2024]. Por ello, es necesario desarrollar un sistema que no solo automatice el pase de lista y mejore la gestión de emergencias, sino que también garantice la seguridad de los datos personales mediante técnicas avanzadas de encriptación.

1.2. Objetivos

Objetivo general

El objetivo general de esta investigación es desarrollar un sistema de pase de lista para la comunidad universitaria de la BUAP que garantice la protección de datos personales sensibles mediante la implementación de un esquema de cifrado híbrido, combinando los algoritmos AES (Advanced Encryption Standard) y RSA (Rivest-Shamir-Adleman), con el fin de elevar el nivel de seguridad en el resguardo de información crítica, especialmente en situaciones de emergencia o siniestros.

Objetivos específicos

- Analizar los fundamentos teóricos y técnicos de los algoritmos de cifrado AES (simétrico) y RSA (asimétrico), identificando sus ventajas y limitaciones en el contexto de la protección de datos.
- Diseñar un algoritmo híbrido que integre las funcionalidades de los algoritmos AES y RSA, optimizando la seguridad y eficiencia en el cifrado de datos sensibles.
- Implementar el algoritmo híbrido propuesto para garantizar la encriptación segura de los datos personales de los usuarios dentro del sistema de pase de lista.

- Desarrollar una plataforma robusta y funcional para el pase de lista que incorpore el sistema criptográfico híbrido, asegurando la integridad, confidencialidad y disponibilidad de la información.
- Realizar pruebas de hacking ético y análisis de vulnerabilidades para evaluar la efectividad del sistema de seguridad implementado y garantizar su resistencia ante posibles ataques.

1.3. Justificación

La automatización del pase de lista mediante el sistema CheckBUAP representa una solución innovadora que no solo optimiza el tiempo en las aulas, sino que además fortalece la seguridad y optimiza la eficiencia en la gestión de emergencias dentro de la BUAP. Este proyecto es relevante porque aborda dos necesidades críticas: la protección de datos personales y la modernización de procesos administrativos y de seguridad.

En primer lugar, al implementar un esquema de encriptación híbrida (AES y RSA), el sistema garantiza la confidencialidad e integridad de la información sensible de los estudiantes, cumpliendo con estándares internacionales de seguridad y normativas locales de protección de datos. Según un informe del Tecnológico Nacional de México, diversas instituciones de educación superior no cuentan con un esquema de seguridad alto, o incluso nulo en ciertas categorías, por lo que no se tiene asegurada una protección de la información total [Mata et al., 2021]. Esto representa un riesgo significativo, ya que, como se ha mencionado anteriormente, el aumento de casos de robo de datos personales aumenta cada año.

En segundo lugar, la funcionalidad de conteo automatizado en tiempo real durante emergencias facilita una respuesta más ágil y eficaz por parte de las autoridades universitarias, lo que puede salvar vidas en situaciones críticas. Aunque actualmente se cuenta con protocolos de seguridad, estos procesos son variados, además de que el personal de seguridad no cuenta con información precisa sobre el número de personas dentro de los edificios, lo que retrasa las operaciones de evacuación y rescate. Con CheckBUAP, se espera reducir este tiempo de respuesta, mejorando significativamente la seguridad de la comunidad universitaria.

Además, este proyecto tiene un impacto social significativo, ya que no solo beneficia a la comunidad universitaria, sino que también establece un precedente en la aplicación de tecnologías avanzadas de encriptación y automatización en el ámbito educativo. La implementación de CheckBUAP contribuye a cerrar esta

brecha, alineándose con las tendencias globales de digitalización y seguridad informática.

Finalmente, este proyecto se alinea con la Ley General de Protección de Datos Personales en Posesión de Sujetos Obligados (LGPDPPSO), que exige a las instituciones implementar medidas de seguridad para el tratamiento de datos personales. Al garantizar la protección de los datos de los estudiantes, CheckBUAP no solo cumple con esta normativa, sino que también fomenta la confianza de la comunidad universitaria en los procesos institucionales.

1.4. Metodología

El desarrollo del sistema CheckBUAP se llevará a cabo siguiendo el Ciclo de Vida del Desarrollo de Software, un marco organizado que garantiza la construcción de software de calidad y seguro, integrando medidas de seguridad en cada fase del proceso y complementándose con una fase adicional de entrenamiento. Esta fase de entrenamiento incluye capacitación en conceptos básicos y avanzados de seguridad informática como fundamentos de criptografía (AES, RSA, cifrado híbrido), buenas prácticas de codificación segura (OWASP Top 10), normativas de protección de datos y técnicas de hacking ético para pruebas de seguridad, lo que permite al equipo contar con los conocimientos necesarios para implementar medidas de seguridad desde las primeras etapas del desarrollo.

En la etapa de planificación se establecen los objetivos, el alcance y los recursos del proyecto, así como los requisitos funcionales (pase de lista automatizado, gestión de emergencias) y no funcionales (seguridad, usabilidad, rendimiento), definiendo el cronograma y los hitos del proyecto. En la fase de análisis se profundiza en los requisitos del sistema y se examinan las necesidades de los usuarios finales (profesores, estudiantes, personal de seguridad), identificando sus necesidades, documentando los casos de uso y los flujos de trabajo del sistema, y definiendo los requisitos de seguridad, como la encriptación de datos y el acceso controlado a la información.

En la etapa de diseño se define la arquitectura del sistema y se planifica su implementación, diseñando la arquitectura del sistema utilizando el patrón Modelo-Vista-Controlador (MVC), creando diagramas UML (casos de uso, secuencia, clases) para modelar el comportamiento del sistema, especificando el esquema de encriptación híbrida (AES y RSA) y su integración en el sistema, y diseñando la interfaz de usuario (UI) para garantizar una experiencia intuitiva y accesible. En la fase de implementación, se desarrolla el sistema según los diseños y es-

pecificaciones definidos, programando las funcionalidades principales, como la generación y lectura de códigos QR, el registro de asistencia y la gestión de emergencias, integrando el esquema de encriptación híbrida para proteger los datos personales, utilizando tecnologías como React para el frontend, Node.js para el backend y MySQL para la base de datos, y aplicando buenas prácticas de codificación segura para prevenir vulnerabilidades.
En la fase de pruebas se valida el funcionamiento del sistema y se asegura su calidad, llevando a cabo pruebas unitarias con el fin de comprobar el adecuado funcionamiento de cada componente , ejecutando pruebas de integración para asegurar que los módulos del sistema trabajen correctamente juntos, llevando a cabo pruebas de seguridad como análisis de vulnerabilidades y hacking ético para garantizar la robustez del sistema y realizando pruebas de usabilidad con usuarios finales para asegurar que la interfaz sea intuitiva y eficiente.
En la fase de despliegue, el sistema se implementa en un entorno real dentro de la BUAP, configurando los servidores y la infraestructura necesaria para el despliegue, migrando los datos existentes (listas de estudiantes, horarios) al nuevo sistema, capacitando a los usuarios finales (profesores, personal de seguridad) en el uso del sistema y monitoreando el sistema durante las primeras semanas para detectar y corregir posibles errores.
Finalmente, en la fase de mantenimiento se garantiza que el sistema funcione de manera óptima y se realizan mejoras continuas, corrigiendo errores reportados por los usuarios, actualizando el sistema para adaptarse a nuevos requisitos o cambios en la normativa de protección de datos, implementando mejoras en la interfaz de usuario y en las funcionalidades del sistema, y realizando auditorías periódicas de seguridad para garantizar la protección continua de los datos.
Esta metodología, basada en el *SDLC (Software Development Life Cycle*, por sus siglas en inglés) y complementada con la fase de entrenamiento, asegura que el sistema CheckBUAP sea desarrollado de manera estructurada, segura y alineada con las necesidades de la comunidad universitaria.

1.5. Alcance esperado

El sistema CheckBUAP tiene como alcance principal la automatización del proceso de pase de lista en la Benemérita Universidad Autónoma de Puebla (BUAP), mejorando la eficiencia en las aulas y proporcionando herramientas para la gestión de emergencias. Se espera que el sistema:

- Reduzca significativamente el tiempo dedicado al pase de lista en las clases.
- Proporcione información en tiempo real sobre la ubicación y número de

personas en los edificios durante emergencias.

- Garantice la seguridad de los datos personales mediante el uso de encriptación híbrida.
- Sea escalable y adaptable para su implementación en otras instituciones educativas.

1.6. Resumen de la estructura de la tesis

Este trabajo de tesis está compuesto por 5 capítulos, los cuales llevan por nombre:

- Capítulo 1: Introducción
- Capítulo 2: Marco teórico
- Capítulo 3: Metodología del desarrollo de la aplicación
- Capítulo 4: Pruebas y resultados
- Capítulo 5: Conclusiones y trabajo a futuro

El primer capítulo es el presentado en esta sección, de la cual se da una breve explicación de los fines del proyecto, así como los objetivos con la finalidad de introducir a los propósitos y lo que se espera alcanzar con ello. El segundo capítulo presenta los conceptos necesarios para comprender a profundidad lo realizado en el proyecto, desde los conceptos matemáticos que utilizan los algoritmos de cifrado, hasta conceptos de seguridad y desarrollo de aplicaciones. El tercer capítulo provee de los pasos realizados en el desarrollo de este proyecto, la creación de la aplicación, así como el tema principal de la tesis, el proceso de cifrado híbrido. El cuarto capítulo cuenta con la redacción de las pruebas realizadas para comprobar la efectividad del cifrado propuesto. Finalmente, el quinto capítulo cuenta con las conclusiones y recomendaciones que se derivan de este trabajo, así como el trabajo esperado.

Capítulo 2

Marco teórico

2.1. Conceptos introductorios matemáticos

2.1.1. Algoritmo de la división

La **división** es una técnica matemática fundamental que asegura que, dados dos números enteros, uno puede ser dividido por otro para obtener un cociente y un residuo. En términos formales, la división se puede describir de la siguiente manera: dados dos enteros a y b, con $b \neq 0$, existen un cociente q y un residuo r, tales que:

$$a = bq + r \quad \text{con} \quad 0 \leq r < |b| \tag{2.1}$$

En la ecuación (2.1) a toma el papel del dividendo y b el papel del divisor. Una vez entendido el concepto anterior, el **algoritmo de la división** es un método para resolver problemas de división, ya sea de números enteros, polinomios o fracciones que parte del siguiente proceso:

Dado un par de números enteros positivos a y b, el objetivo del algoritmo es hallar enteros q y r que satisfagan las condiciones $a = bq + r$ y $0 \leq r < b$. Esto se obtiene al sustraer b repetidamente de a hasta que el resultado sea menor que b pero permanezca sin ser negativo.

$$0 \leq a - b - b - b - ... - b = a - bq < b$$

El número total de $b's$ que sean sustraídos es el cociente q. La cantidad $a - bq$ es igual al residuo r.

- La entrada consta de, a un entero no negativo y b un entero positivo.
- El algoritmo comienza con $r = a$ y $q = 0$.
- Repetidamente sustrae b de r hasta que un número menor que b sea obtenido. Agrega 1 a q cada vez que b sea sustraído.
- **mientras** $r \geq b$
  $r = r - b$
  $q = q + 1$
  finalizar **mientras**
- Después de la ejecución del ciclo **mientras**, $a = bq + r$
- La salida consta de q y r, enteros no negativos.

Ejemplo de aplicación del algoritmo de división

Considere el algoritmo de división entera descrito anteriormente. Para ilustrar su funcionamiento, se presenta un ejemplo con valores específicos.

- **Datos de entrada**
  $a = 17$ (un entero no negativo, el dividendo)
  $b = 5$ (un entero positivo, el divisor)
- **Inicialización**
  El algoritmo comienza inicializando las variables:
  $r = a = 17$ (el residuo inicial es igual al dividendo)
  $q = 0$ (el cociente inicial es cero)
- **Ciclo Mientras**
  El ciclo **mientras** se ejecuta mientras el residuo r sea mayor o igual que el divisor b. En cada iteración, se realiza lo siguiente: se resta b de r y se incrementa q en 1. A continuación, se detalla la ejecución del ciclo:

  Iteración 1:

  $$r = 17 - 5 = 12$$

  $$q = 0 + 1 = 1$$

  Iteración 2:

  $$r = 12 - 5 = 7$$

  $$q = 1 + 1 = 2$$

Iteración 3:

$$r = 7 - 5 = 2$$

$$q = 2 + 1 = 3$$

En la tercera iteración, el residuo $r = 2$ es menor que el divisor $b = 5$. Por lo tanto, el ciclo **mientras** finaliza.

- **Resultados**
  Al finalizar el algoritmo, se obtienen los siguientes resultados:
  $q = 3$ (el cociente de la división)
  $r = 2$ (el residuo de la división)

Para garantizar la veracidad del algoritmo, se verifica que se cumple la relación fundamental de la división entera: $a = bq + r$. Sustituyendo los valores obtenidos:

$$17 = 5 \cdot 3 + 2$$

y simplificando:

$$17 = 15 + 2 \quad \Rightarrow \quad 17 = 17$$

La igualdad se cumple, lo que confirma que los resultados son correctos.
Este ejemplo demuestra cómo el algoritmo realiza la división entera de manera eficiente, obteniendo el cociente q y el residuo r que satisfacen la relación $a = bq + r$.

2.1.2. Máximo común divisor

El **máximo común divisor (MCD)** de dos números a y b, es el mayor número entero que divide a ambos sin dejar residuo. Si d es el MCD de a y b, entonces cumple con lo siguiente:

1. $d \mid a$ y $d \mid b$, es decir, d divide a a y b sin dejar residuo.
2. Si d divide a a, es decir $d \mid a$, entonces existe un entero k tal que $a = d \cdot k$.
3. Si existe otro divisor común d' de a y b, entonces $d' \leq d$, lo que se traduce en que d es el mayor de todos los divisores comunes.

Considere que a y b, toman los valores de 18 y 24 respectivamente, los divisores para cada uno son:

- Para 18: $\{1, 2, 3, 6, 9, 18\}$
- Para 24: $\{1, 2, 3, 4, 6, 8, 12, 24\}$

Los divisores comunes de 18 y 24 son $1, 2, 3, 6$, y el mayor de ellos es 6, por lo tanto, el $MCD(18, 24) = 6$.

2.1.3. Algoritmo de Euclides

Antes de comenzar a hablar del algoritmo, es importante mostrar el siguiente teorema, el cual tiene relevancia para el proceso y se origina a partir de la definición de división 2.1, ya que de acuerdo a los términos presentados, b es un divisor de a o que a es múltiplo de b [Carracedo and Alix, 1992].

Teorema 1. *Dados estos, sean $a, b, c \in \mathbb{Z}$ con $a \neq 0$, si $b \mid a$ y $b \mid c$, entonces*

$$b \mid (ma + nc) \tag{2.2}$$

con $m, n \in \mathbb{Z}$.

Una vez expuesto el teorema anterior, el **algoritmo de Euclides** se describe como un método eficiente para determinar el MCD de dos números a y b que parte del siguiente proceso:

1. Sean a y b enteros que cumplen con la condición $a > b \geq 0$.

2. Para encontrar el MCD de a y b, primero se comprueba que $b = 0$. Si lo es, entonces $MCD(a, b) = a$. Si no lo es, entonces $a > 0$ y se puede cumplir la siguiente condición:

   $$a = bq + r, \tag{2.3}$$

   donde

   $$0 \leq r < b$$

   por lo tanto se cumple que $MCD(a, b) = MCD(b, r)$. Esto se debe a que si $d = MCD(a, b)$ entonces, d divide tanto a a como a b. Aplicando el teorema 1 como $d \mid a$ y $d \mid b$, se deduce que $d \mid (a - bq)$, es decir, $d \mid r$. Por lo tanto, d es un divisor común de b y r. Además cualquier divisor común de b y r también divide a $a = bq + r$, lo que garantiza que $MCD(a, b) = MCD(b, r)$.

3. Ahora repite el proceso a partir del punto 2, pero en lugar de a, usa b; y en lugar de b, usa r. Las repeticiones aseguran que el proceso terminará con $r = 0$ porque cada nuevo residuo será menor que el anterior y ninguno es negativo. Cuando $r = 0$, el algoritmo termina y el MCD es el último residuo no nulo.

Ejemplo del algoritmo de Euclides

Por ejemplo, considere encontrar el MCD de 252 y 105, donde $a = 252$ y $b = 105$; siguiendo el algoritmo presentado anteriormente:

1. Sean $a = 252$ y $b = 105$. Comenzamos verificando que $b = 0$. Como no lo es, procedemos a aplicar la división:

   $$252 = 105 \cdot 2 + 42$$

   Aquí, el residuo $r = 42$, por lo tanto, ahora tenemos que encontrar el MCD de 105 y 42.

2. Repetimos el proceso usando $a = 105$ y $a = 42$. Realizamos la división:

   $$105 = 42 \cdot 2 + 21$$

   Aquí, el residuo $r = 21$, por lo que ahora tenemos que encontrar el MCD de 42 y 21.

3. Continuamos usando $a = 42$ y $a = 21$. Realizamos la división:

   $$42 = 21 \cdot 2 + 0$$

   El residuo ahora es $r = 0$, por lo que el proceso ha terminado y el MCD es el último divisor no nulo, es decir:

   $$MCD(252, 105) = 21$$

Por lo tanto, el MCD de 252 y 105 es 21.

2.1.4. Primos relativos

Teniendo en cuenta el concepto de MCD, se considera que dos enteros a y b son primos relativos, si y solo si $MCD(a, b) = 1$. Es decir, no tienen ningún otro divisor común aparte del 1. Por ejemplo, teniendo $a = 8$ y $b = 15$, los divisores de cada uno son:

- Para 8: $\{1, 2, 4, 8\}$
- Para 15: $\{1, 3, 5, 15\}$

El único divisor común entre 8 y 15 es 1, por lo que, $MCD(8, 15) = 1$, lo que significa que son *primos relativos*.

2.1.5. Mínimo común múltiplo

El **mínimo común múltiplo (MCM)** de dos números a y b, es el menor número entero positivo que es múltiplo de ambos números a y b. Si m es el MCM de a y b, entonces cumple con lo siguiente:

1. $a \mid m$ y $b \mid m$, es decir, m es divisible por a y b.
2. Si existe otro múltiplo común m' de a y b, entonces $m' \geq m$, lo que se traduce a que m es el menor de todos los múltiplos comunes.

Se propone que $a = 12$ y $b = 15$, algunos de los múltiplos de cada uno son:

- Para 12: $\{12, 24, 36, 48, 60, ...\}$
- Para 15: $\{15, 30, 45, 60, 75, ...\}$

El menor múltiplo común de 12 y 15 es 60, por lo tanto, el $MCM(12, 15) = 60$.

2.1.6. Aritmética modular

La aritmética modular es un sistema numérico utilizado para trabajar con congruencias y restos. En lugar de trabajar con números en una secuencia infinita, en la aritmética modular se consideran equivalentes aquellos números que tienen el mismo residuo cuando se dividen por un número llamado **módulo**. Formalmente, sean m y n enteros y d un entero positivo, se dice que m es congruente a n módulo d denotado por

$$m \equiv n \pmod{d} \tag{2.4}$$

Esto significa que m y n dejan el mismo residuo al dividirse entre d. Equivalentemente, existe un $k \in \mathbf{Z}$ tal que:

$$m - n = kd \tag{2.5}$$

Propiedades de la aritmética modular

La aritmética modular cuenta con las siguientes propiedades esenciales:

- **Propiedad conmutativa:** La suma y la multiplicación en aritmética modular son conmutativas, es decir, $m + n \equiv n + m \pmod{d}$ y $m \cdot n \equiv n \cdot m \pmod{d}$.
- **Propiedad asociativa:** Tanto la suma como la multiplicación son asociativas: $(m+n)+p \equiv m+(n+p) \pmod{d}$ y $(m \cdot n) \cdot p \equiv m \cdot (n \cdot p) \pmod{d}$.

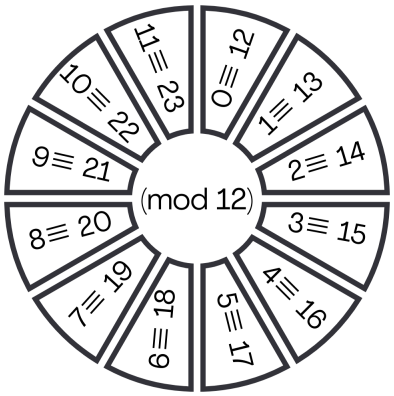

Figura 2.1: Representación de un reloj en módulo 12

- **Elemento neutro:** Existe un elemento neutro tanto para la suma como para la multiplicación. El elemento neutro para la suma es 0, es decir, $m + 0 \equiv m \pmod{d}$, y para la multiplicación es 1, es decir, $m \cdot 1 \equiv m \pmod{d}$.
- **Elemento inverso:** En la suma, todo número m tiene un inverso aditivo $-m$, tal que $m + (-m) \equiv 0 \pmod{d}$. En cuanto a la multiplicación, un número m tiene inverso multiplicativo si y solo si $MCD(m, d) = 1$.
- **Distributividad:** La multiplicación es distributiva respecto a la suma, es decir, $m \cdot (n + p) \equiv (m \cdot n) + (m \cdot p) \pmod{d}$.

Ejemplo de aritmética modular

Un ejemplo común de aplicar en la aritmética modular tiene que ver con los relojes. Prácticamente, el uso del reloj se distingue por tener ciclos, es decir, el tiempo no comenzó desde el segundo 0 y ha seguido así consecutivamente. Lo que hacemos al llegar al final del día, a las 23:59 horas, es "reiniciarse" a las 00:00. Esto se debe a que gira en torno al módulo 12, como se observa en la Figura 2.1.

Se puede notar que, cada representación es equivalente a la hora que se maneja en el día a día. Por ejemplo, $0 \equiv 12 \pmod{12}$ se traduce a que 0 y 12 dejan el mismo residuo al dividirlos entre 12. Al comprobarlo mediante el algoritmo de la división, por lo que $a = 12$ y $b= 12$, donde se busca el residuo

r, por lo que

$$12 = 12(1) + 0$$

el cociente es $q = 1$ y el residuo es $r = 0$. De esta manera comprobamos que el residuo de $12 \mid 0$ es igual al de $12 \mid 12$, ambos tienen residuo 0, siendo la equivalencia del reloj correcta.

2.1.7. Teorema de Bézout

El algoritmo de Euclides proporciona otros usos más allá de solo encontrar una manera más fácil de encontrar el máximo común divisor (MCD) de dos números. También indica la relación entre dos números y su MCD, esta relación es llamada **Identidad de Bézout**, la cual indica que si el máximo común divisor de a y b es d, entonces

$$d = ar + bs \tag{2.6}$$

para algunos enteros r y s. En otras palabras, el teorema dice que el $MCD(a, b)$ se puede expresar como una combinación lineal de a y b.

Ejemplo del teorema de Bézout

Supongamos que queremos expresar el MCD de 252 y 105 como una combinación lineal de estos números. Sabemos que:

$$MCD(252, 105) = 21$$

Siguiendo los pasos del algoritmo de Euclides en la ecuación 2.3, encontramos que:

1. Primero, se divide 252 entre 105:

$$252 = 105 \cdot 2 + 42$$

2. Luego, se divide 105 entre 42:

$$105 = 42 \cdot 2 + 21$$

3. Finalmente, se divide 42 entre 21:

$$42 = 21 \cdot 2 + 0$$

Como el último residuo no nulo es 21, el MCD de 252 y 105 es 21.

Ahora, se retrocede para expresar 21 como una combinación lineal de 252 y 105:

1. A partir de la segunda ecuación:

$$21 = 105 - 42 \cdot 2$$

2. Luego, se sustituye 42 de la primera ecuación:

$$42 = 252 - 105 \cdot 2$$

Sustituyendo esto en la expresión anterior:

$$21 = 105 - (252 - 105 \cdot 2) \cdot 2$$

Simplificando:

$$21 = 105 - 2 \cdot 252 + 4 \cdot 105$$

$$21 = 5 \cdot 105 - 2 \cdot 252$$

Por lo tanto, hemos expresado 21 como una combinación lineal de 252 y 105:

$$21 = 5 \cdot 105 - 2 \cdot 252$$

En este caso, los coeficientes de Bézout son $r = -2$ y $s = 5$.

2.1.8. Pequeño teorema de Fermat

El **pequeño teorema de Fermat** es el nombre que recibe el último teorema de Fermat, el cual proporciona la teoría crucial para el algoritmo de criptografía RSA (Rivest, Shamir y Adleman).

Teorema 2. *Si p es cualquier número primo y a es un entero no divisible por p entonces,*

$$a^{p-1} \equiv 1 \mod p \tag{2.7}$$

En términos de divisibilidad se puede decir que, si p es un número primo y a es coprimo de p, entonces p divide $a^{p-1} - 1$.

2.1.9. Función ϕ de Euler

La **función ϕ de Euler** es una función que permite determinar cuántos números enteros positivos menores o iguales que n son coprimos con n. Es decir,

si $n \in \mathbb{Z}$ y $n > 0$, $\phi(n)$ es el número de enteros k tales que $1 \leq k \leq n$ y $MCD(k, n) = 1$. La cual se escribe de manera generalizada como,

$$\phi(n) = \text{Cantidad de enteros } k \text{ tales que } 1 \leq k \leq n \text{ y } MCD(k, n) = 1$$

‘

Al igual que es importante definir que, para un número entero n dado, su factorización prima es:

$$n = p_1^{k_1} p_2^{k_2} ... p_r^{k_r} \tag{2.8}$$

donde $p_1, p_2..., p_k$ son primos distintos y $k_1, k_2, ..., k_r$ son sus exponentes. Antes de pasar a la definición de la función ϕ de Euler, se mencionarán algunas propiedades que dan paso a la misma.

Propiedades de la función ϕ

- **Para un número primo p:** Si $n = p^k$ donde p es un número primo y $k \geq 1$, entonces:

$$\phi(p^k) = p^k \left(1 - \frac{1}{p}\right) = p^k - p^{k-1}$$

- **Función multiplicativa:** La función ϕ es **multiplicativa**, lo que significa que si a y b son coprimos (es decir, $MCD(a, b) = 1$), entonces:

$$\phi(ab) = \phi(a)\phi(b)$$

- **Relación con primos:** Si p es un número primo, entonces:

$$\phi(p) = p - 1$$

Usando las propiedades anteriores, se puede calcular $\phi(n)$ para cualquier número entero positivo n, teniendo en cuenta lo siguiente:

1. Se escribe la forma factorizada de n en primos como en la forma 2.8.

2. Como la función de Euler es multiplicativa para números coprimos, se puede escribir de la siguiente manera:

$$\phi(n) = \phi(p_1^{k_1}) \cdot \phi(p_2^{k_2}) \cdot ... \cdot \phi(p_r^{k_r})$$

3. Usando la propiedad "Para un número primo p", se transforma a:

$$\phi(p_i^{k_i}) = (p_i^{k_i}) - (p_i^{k_i-1}) = (p_i^{k_i})\left(1 - \frac{1}{p_i}\right)$$

4. Finalmente multiplicando los valores de $\phi(p_i^{k_i})$ para obtener $\phi(n)$:

$$\phi(n) = \prod_{i=1}^{r} \phi(p_i^{k_i}) = \prod_{i=1}^{r} \phi p_i^{k_i}\left(1 - \frac{1}{p_i}\right)$$

La cual se puede expresar de la siguiente manera:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)...\left(1 - \frac{1}{p_k}\right) \tag{2.9}$$

[García et al., 2002].

Teorema de Euler

El **teorema de Euler** constituye una extensión del pequeño teorema de Fermat, y establece lo siguiente:

Teorema 3. *Sea n un entero positivo. Si a es un entero coprimo con n es decir,* $MCD(a, n) = 1$*, entonces,*

$$a^{\phi(n)} \equiv 1 \pmod{n} \tag{2.10}$$

2.2. Introducción criptografía

Se han mostrado diversos conceptos matemáticos utilizados por el algoritmo criptográfico de RSA y el algoritmo de hash de AES, por lo cual es importante comprender a grandes rasgos qué implica cada uno de ellos sin importar la elección de un tipo de algoritmo específico.

La criptografía es usada para proteger la información manteniendo su significado o contenido secreto y haciéndolo incomprensible para cualquiera que no tenga una manera de desencriptar la información protegida. El objetivo de todo sistema de encriptación es transformar un conjunto de datos original, llamado **texto plano** o en inglés *plaintext*, a una forma encriptada incomprensible llamada **texto cifrado** o en inglés *ciphertext*. [ISC2, 2022]

Propiamente utilizados, solos o en combinación, proveen un rango de servicios que pueden ayudar a proveer posturas de seguridad de muchas maneras:

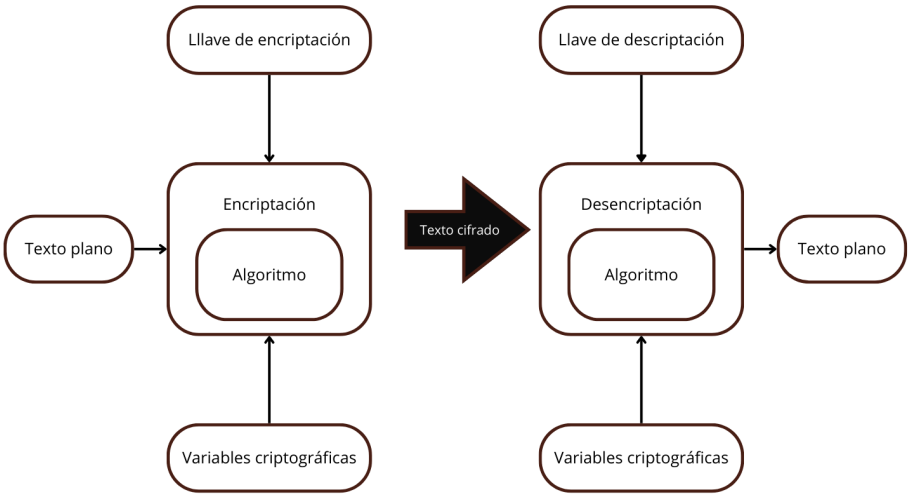

Figura 2.2: Modelo del sistema de encriptación elaborado con material de ISC2

- *Confidencialidad:* Provee confidencialidad al esconder el mensaje para que de esta manera no pueda ser entendido por nadie excepto por el destinatario. La confidencialidad mantiene la información secreta a todos aquellos que no son autorizados para tenerlo.
- *Integridad:* Las funciones hash y las firmas digitales pueden proveer servicios de integridad para permitir al destinatario de un mensaje verificar que no ha sido modificado por malicia o error.

Un **sistema de encriptación**, como el que se detalla en la Figura 2.2, es un conjunto de hardware, software, parámetros de control y métodos operacionales que proveen un conjunto de servicios de encriptación. El **texto plano** son los datos o mensajes en su forma y formato sin encriptar. El texto plano puede ser:

- Imágenes, audio y vídeo en bruto o en su forma comprimida.
- Textos legibles para humanos y datos numéricos, con o sin elementos de lenguaje de marcado y metadata.
- Archivos de bases de datos o registros y campos de una base de datos.
- Cualquier otra cosa que pueda ser representada en una forma digital para procesamiento de computadora, transmisión y almacenamiento.

Se distinguen dos tipos de encriptación: simétrica y asimétrica; se hablará sobre estos temas en las posteriores secciones.

2.2.1. Encriptación simétrica

La encriptación simétrica es un método que emplea una única clave para cifrar y descifrar la información. Tanto el remitente como el destinatario deben contar con la misma clave; se podría decir que el proceso de descifrado es el proceso espejo del de cifrado. [ISC2, 2022]

Funcionamiento

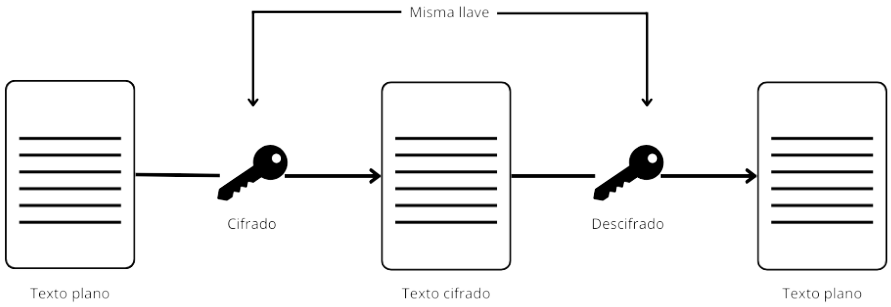

Figura 2.3: Funcionamiento de la encriptación simétrica

Como se ha mencionado, al tener uso de una misma llave, es necesario que los involucrados compartan la misma llave. El emisor utiliza la clave pública del receptor para cifrar el *plaintext* y el receptor emplea la misma clave para descifrar el *ciphertext* y recuperar el texto plano; observe el procedimiento en la figura 2.3. Un algoritmo comúnmente utilizado es el **AES (Advanced Encryption Standard)**. En la figura 2.4 se observa un ejemplo del proceso de AES.

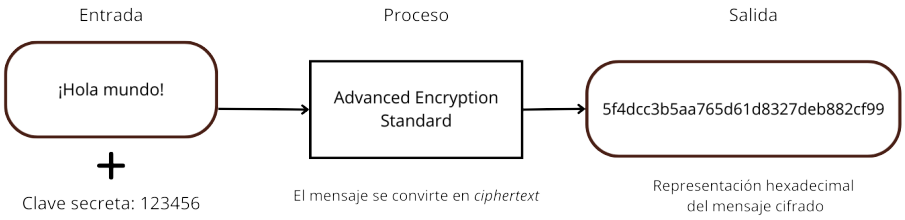

Figura 2.4: Ejemplificación del funcionamiento de encriptación simétrica

2.2.2. Encriptación asimétrica

En contraste con la encriptación simétrica, la asimétrica emplea una llave para encriptar y otra para desencriptar la entrada del texto plano, lo que genera un fuerte contraste en temas de seguridad en comparación con la encriptación simétrica.

Funcionamiento

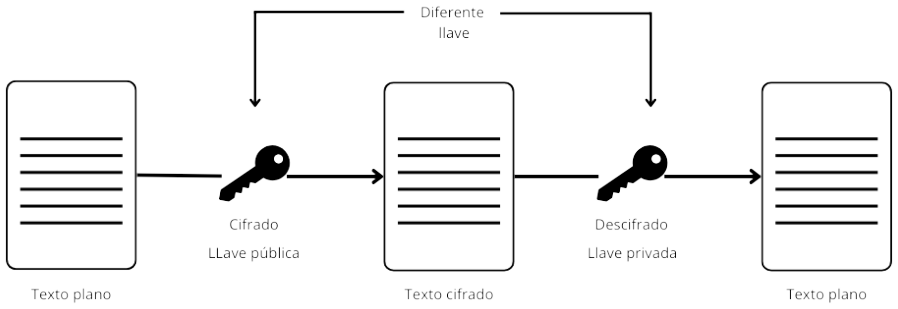

Figura 2.5: Funcionamiento de la encriptación asimétrica

Un usuario que desea comunicarse mediante un algoritmo asimétrico debe generar un par de llaves. Para garantizar una generación fuerte, este proceso suele realizarse a través de una aplicación criptográfica o una infraestructura de clave pública (**PKI, por sus siglas en inglés *Public Key Infrastructure***), lo que elimina la necesidad de manejo directo por parte del usuario. Una de las llaves, conocida como **clave privada**, se mantiene en secreto, mientras que la otra, llamada **clave pública**, puede compartirse libremente. Este proceso se ilustra en la figura 2.5.[ISC2, 2022] Un algoritmo popular para este propósito es **RSA (Rivest–Shamir–Adleman)**, cuyo funcionamiento se presenta en la figura 2.6.

2.2.3. Comparación

Ambos tipos de criptografía cuentan con sus beneficios y contras, los cuales se visualizan en la tabla comparativa 2.1.

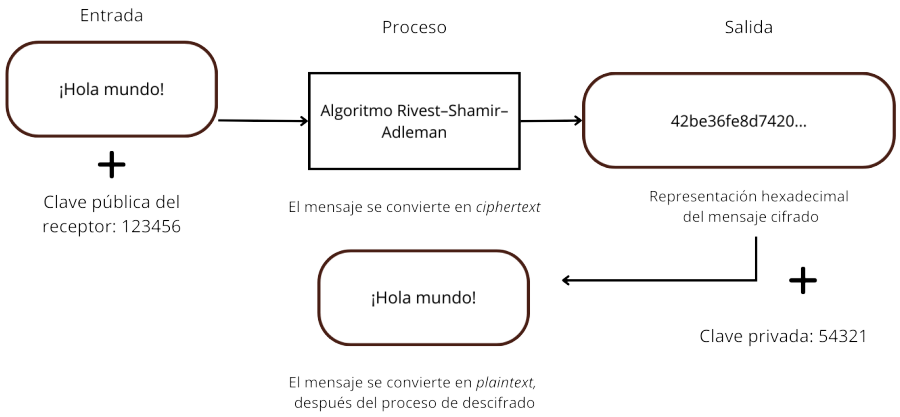

Figura 2.6: Ejemplificación del funcionamiento de encriptación asimétrica

Tipo	Simétrico	Asimétrico
Velocidad	Cifrado rápido	Cifrado lento
Clave	Única clave	Clave pública y privada
Tamaño claves	128 o 256 bits	2048 bits o más
Riesgos	Robo de la clave secreta	Robo de la clave privada
Uso	Cifrar y transferir grandes cantidades de datos	Cifrar y transferir grandes cantidades de datos

Cuadro 2.1: Comparativa entre cifrado simétrico y asimétrico elaborado con material de Socas Gutiérrez [Socas Gutiérrez and Déniz, 2023]

2.3. Algoritmo RSA (Rivest-Shamir-Adleman)

2.3.1. Historia y definición del algoritmo RSA

La distribución de claves se reconoce como uno de los principales desafíos que enfrentan los algoritmos de cifrado, puesto que, sea cual sea la situación, existe la posibilidad de que la clave sea robada, ya que es necesaria su distribución para desencriptar. Tomando en cuenta estas necesidades, Diffie y Hellman en 1973 proponen un nuevo modelo de criptosistema, el cual proponía que las claves de desencriptación y encriptación fueran tan diferentes que no era posible obtener una a partir de la otra. Es por esto que proponen un algoritmo que debía cumplir con tres requerimientos, en donde se tiene un algoritmo de encriptación con clave E y otro de desencriptación con clave D:

1. $D(E(P))) = P$
2. Sea sumamente difícil deducir D a partir de E.
3. E no puede descifrarse a través de un ataque de texto plano seleccionado.

De manera general, los primeros dos puntos expresan que si aplicamos la clave D a un mensaje cifrado, $E(P)$ podremos obtener un mensaje P en texto plano. El tercer punto explica que, sea cual sea la situación y las experimentaciones que tengan con el algoritmo, no existiría razón para que una clave de encriptación sea pública. Esta propuesta llevó a la creación de diversos algoritmos que cumplan con las características antes mencionadas [Tanenbaum, 2003]. En el año 1978, Ron Rivest, Adi Shamir y Leonard Adleman crearon el algoritmo RSA (por las iniciales de sus creadores) en el MIT (Massachusetts Institute of Technology) el cual durante más de 30 años se ha considerado muy sólido y su principal oponente se basa en la computación cuántica y sus alcances hasta el momento [Julio Mulero and (EDS.), 2024].

Utilizando los conceptos descritos en la sección 2.1, y suponiendo que A quiere mandar un mensaje cifrado a B , el algoritmo de RSA se ejecuta de la siguiente manera:

Generación de claves

1. A escoge un par de números primos muy grandes y no públicos, p y q tales que se obtiene

   $$n = p \cdot q$$

   Aquí, n será el módulo del sistema y determinará el tamaño del espacio de mensajes posibles.

2. A calcula $\phi(n)$, donde $\phi(n) = (p-1)(q-1)$. Esta función, conocida como la función de Euler, cuenta cuántos números son coprimos con n.

3. A escoge un número entero e que sea primo relativo con $\phi(n)$, es decir, $MCD(e, \phi(n)) = 1$. Para verificar esto, se utiliza el algoritmo de Euclides (2.3), que permite calcular eficientemente el máximo común divisor (MCD) utilizando el algoritmo de la división 2.1.

4. A encuentra un número d tal que

   $$e \cdot d \equiv 1 (mod\ \phi(n))$$

   Este valor d se obtiene encontrado el inverso modular de e respecto a $\phi(n)$, utilizando el teorema de Bézout 2.6. Este teorema establece que si

$MCD(e, \phi(n) = 1$ entonces existen enteros d y k tales que:

$$(e \cdot d) + (k \cdot \phi(n) = 1$$

Resolviendo esta ecuación con el algoritmo de Euclides, se obtiene d.

5. La clave pública de A es (e, n), la cual se distribuye libremente. La clave privada de A es (d, n), que debe mantenerse en secreto.

Cifrado del mensaje por B

6. B desea enviar un mensaje M a A. Si M es un texto, se convierte en un número utilizando una codificación como ASCII o Unicode.

7. B utiliza la clave pública (e, n) de A para cifrar el mensaje M. El mensaje cifrado C se calcula como:

$$C \equiv M^e \pmod{n}$$

Aquí la aritmética modular 2.5 juega un papel clave, ya que de la operación de exponenciación se realiza en módulo n.

8. Para evitar cálculos innecesarios con números extremadamente grandes, se utiliza la exponenciación modular rápida. Este método emplea el algoritmo de la división en cada paso para reducir los exponentes mediante sucesivas divisiones:

$$a^b \mod m = \left(\left(a^{b/2} \mod m\right) \cdot \left(a^{b/2} \mod m\right)\right) \mod m$$

Descifrado del mensaje por A

9. A emplea su clave privada (d, n) para descifrar el mensaje cifrado C. El mensaje original M' se recupera como:

$$M' \equiv C^d \pmod{n}$$

10. Dado que $e \cdot d \equiv 1 \pmod{\phi(n)}$, por el teorema de Euler 2.10 se cumple:

$$M^{\phi(n)} \equiv 1 \pmod{n}$$

Esto garantiza que el mensaje original se recupere correctamente:

$$M' \equiv M$$

11. El pequeño teorema de Fermat 2.7 también respalda la validez del descifrado. Este teorema establece que para un número primo p y un número primo a coprimo con p:

$$a^{p-1} \equiv 1 \pmod{p}$$

Este principio, generalizado por el teorema de Euler, asegura la elección correcta de exponentes en RSA [de Tejada and Manzano, 2023].

2.4. Algoritmo AES (Advanced Encryption Standard)

2.4.1. Estructura de cifrado Feistel

Para comprender el algoritmo AES, es importante mencionar que muchos de los algoritmos de cifrado simétrico por bloques siguen la estructura propuesta por Horst Feistel en 1973. En estos algoritmos, la entrada consiste en un bloque de texto claro de tamaño *2w* bits y una clave K. El bloque de texto claro se divide en dos mitades, L_0 y R_0, las cuales atraviesan n etapas de procesamiento antes de combinarse nuevamente para generar el bloque de texto cifrado. Cada etapa i tiene como entrada L_{i-1} y R_{j-1}, que se generan de la etapa $n-1$, al igual que una subclave K_i generada de K; las claves K_i son diferentes a K y entre ellas mismas, ya que se generan a partir de la clave mediante un algoritmo de generación de subclaves. Todas las etapas cuentan con una estructura similar: se genera una sustitución sobre la mitad izquierda de los datos, la cual se crea aplicando una *función de etapa F* a la mitad derecha de los datos y haciendo luego una operación XOR a la salida de la función y la mitad izquierda de los datos. La función de etapa, en general, cuenta con la misma estructura para cada etapa, pero tiene como parámetro la subclave de etapa K_i. Posterior a esta sustitución, se lleva a cabo una permutación, en la cual se intercambian las dos mitades de los datos. [Stallings, 2004]

Esta estructura era utilizada por el predecesor del algoritmo AES: DES (Data Encryption Standard), un algoritmo que en la actualidad se considera inseguro, y en su apogeo, se preveía que sería reemplazado, ya que es un algoritmo considerablemente lento en su implementación en software, ya que fue diseñado para

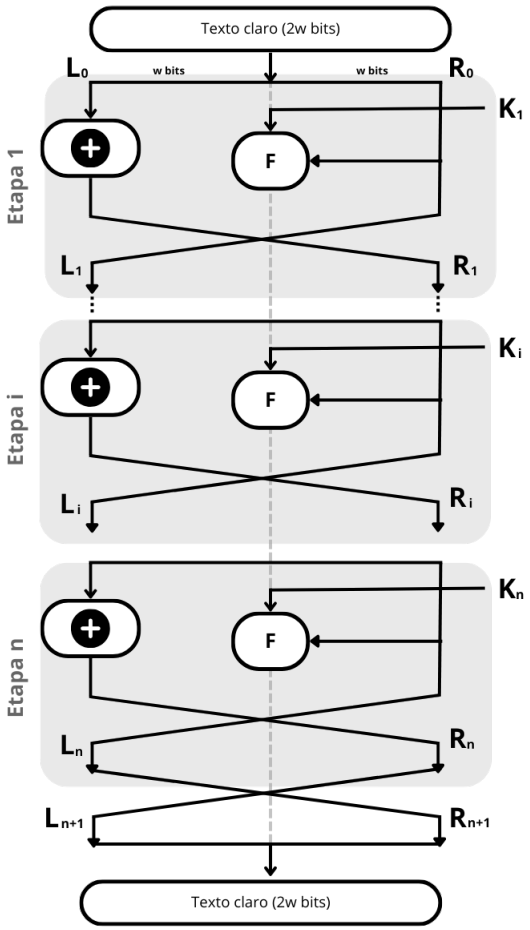

Figura 2.7: Demostración de red clásica Feistel elaborado con material de Stallings [Stallings, 2004]

el cifrado de hardware; por este motivo, la versión más segura de DES, 3DES, era tres veces más lenta. Además de contar con un tamaño de bloque de 64 bits, el cual no contaba con la seguridad necesaria en ese momento ni ahora, todo esto llevó a la necesidad de crear un sucesor. [Stallings, 2004]

2.4.2. Historia y definición del algoritmo AES

El algoritmo AES, por sus siglas en inglés *Advanced Encryption Standard*, es el sucesor del algoritmo DES, *Data Encryption Standard*, presentado por el Instituto Nacional de Estándares y Tecnología, NIST, por sus siglas en inglés, en el año 2001. También es conocido como "Rijndael" por la combinación de los apellidos de sus creadores, Jon Daemen y Vincent Rijmen. El algoritmo utiliza el campo Galois para llevar a cabo sus operaciones matemáticas, el cual consiste en una construcción matemática donde las operaciones de adición, sustracción, multiplicación y división se redefinen y se cuenta con un número limitado de enteros; en el caso del algoritmo AES solo permite un número de 8 bits (un número entre 0 y 255) y todas sus operaciones matemáticas resultarán en un número de 8 bits. En AES, la suma y la resta representan operaciones de tipo XOR, y no hay una diferencia entre la adición y la sustracción. [Maiorano, 2009]

2.4.3. Explicación del algoritmo AES

Una de las principales diferencias de AES es que su estructura no está compuesta por una estructura Feistel, en la cual la mitad del bloque de datos se usaba para cambiar la otra mitad, y luego intercambiarse entre sí (Figura 2.7); AES utiliza todo el bloque de datos en paralelo en cada etapa, lo que permite que su procesamiento sea más rápido. AES opera en bloques de 128 bits usando tres diferentes tamaños de bits de las claves: 128, 192 y 256 bits; así como, a diferencia de DES, la unidad de procesamiento de AES es el byte, a diferencia de DES, cuya unidad de procesamiento es el bit. Al igual que DES, AES utiliza un cifrado de bloque y su proceso sigue una secuencia de N etapas, las cuales dependen de la longitud de la llave: para una llave de 128 bits se necesitan 10 rondas, para una llave de 192 bits, 12; y para una llave de 256 bits se utilizan 14 etapas; de esta manera, dependiendo de la longitud de la llave se le denomina al algoritmo: AES-128, AES-192 o AES-256. [Mammeri, 2024]

Para la explicación de este algoritmo, tomaremos el ejemplo de una longitud de clave de 128 bits, es decir, AES-128. Este bloque de 128 bits de entrada se representa como una matriz cuadrada de bytes, la cual se copia en el vector *Estado*, que se modifica en cada etapa de cifrado o descifrado; al finalizar el proceso, el vector *Estado* se copia en una matriz de salida. Al igual que el bloque de entrada, la clave de 128 bits se representa como una matriz cuadrada de bytes; luego, esta clave se expande en un vector de palabras para poder generar las claves, cada palabra se compone de 4 bytes, y el número total consta de 44 para

la clave de 128 bits. La disposición de los bytes dentro de la matriz, que lleva por nombre *in*, se establece por columnas. Dentro del proceso se encuentran 4 funciones principales, de las cuales 3 son de sustitución y una de permutación:

- **Sustitución de bytes**: esta función utiliza una tabla denominada **caja** S^4, la cual se encarga de realiza una sustitución byte a byte del bloque.
- **Mezcla de columnas**: esta función realiza una sustitución que altera cada byte de una columna en función de todos los bytes de la columna.
- **Suma de la clave de etapa**: realiza una operación XOR bit a bit del bloque actual con una parte de la clave extendida, utilizando la propiedad de $A \oplus B \oplus A = B$.
- **Desplazamiento de filas**: esta función realiza una permutación fila por fila. [Stallings, 2004]

En la figura 2.8 se detalla la aplicación de las funciones anteriores para un proceso de AES-128, la cual cuenta con 9 etapas con 4 de estas funciones y la décima de tres fases. Únicamente la función de suma de la clave de etapa emplea la clave, es por este motivo que van al principio y final, ya que cualquier otra fase aplicada en estos lugares sería reversible y, por lo tanto, insegura. La función de suma de la clave no funcionaría sola, ya que las otras tres fases son las encargadas de desordenar los bits, pero al no utilizar la clave, no proporciona seguridad por sí mismas. Se puede observar en la figura 2.8 que el proceso de cifrado no es idéntico al de descifrado, y las últimas etapas de estos procesos solo cuentan con tres fases, lo cual es una particularidad del algoritmo de AES y es necesario para que el cifrador sea reversible. [Stallings, 2004]

2.5. Conceptos de desarrollo y diseño de software

2.5.1. Ciclo de vida del software

El Ciclo de Vida del Desarrollo de Software (SDLC, por sus siglas en inglés, *Software Development Life Cycle*) representa un marco que determina las fases necesarias para construir software de calidad, que además, visto desde una perspectiva de seguridad, ayuda a garantizar la seguridad de sus aplicaciones a través de la aplicación de un conjunto de medidas a nivel de desarrollo y, en el caso de organizaciones, en las prácticas de seguridad relativas a ellas. De manera general, muchos autores proponen diversas fases de este modelo, en las cuales, en algunas se llega a un consenso; estas fases incluyen la planificación, análisis, diseño, implementación, pruebas, despliegue y mantenimiento (Figura 2.9). Cada

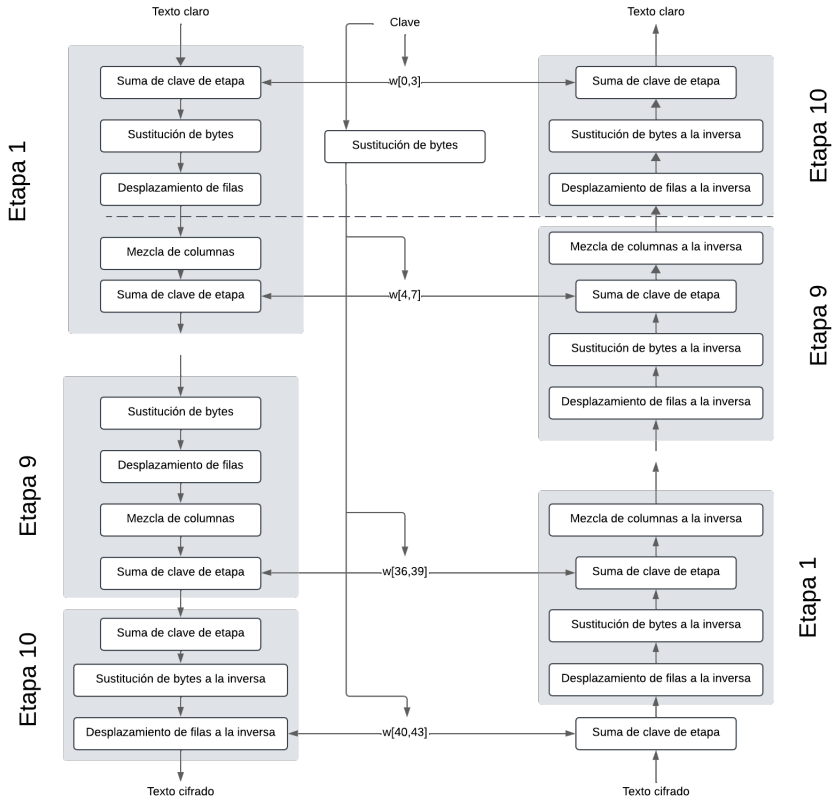

Figura 2.8: Proceso de cifrado y descifrado de AES elaborado con material de Stallings [Stallings, 2004]

fase tiene un objetivo específico: la planificación establece los recursos y el alcance del proyecto; el análisis define los requisitos; el diseño estructura la solución; la implementación construye el software; las pruebas validan su funcionamiento; el despliegue lo pone en producción; y el mantenimiento corrige errores y mejora el sistema. Este enfoque estructurado garantiza que el software satisfaga tanto los estándares de calidad como los requisitos del usuario [Kendall et al., 2005]. Es importante mencionar que a este enfoque se agrega una propuesta de otro modelo con una fase adicional previa a todas las anteriores: entrenamiento. Esta fase se refiere a la formación previa acerca de los conceptos básicos y las últimas tendencias en seguridad y privacidad de la información [Candel, 2020].

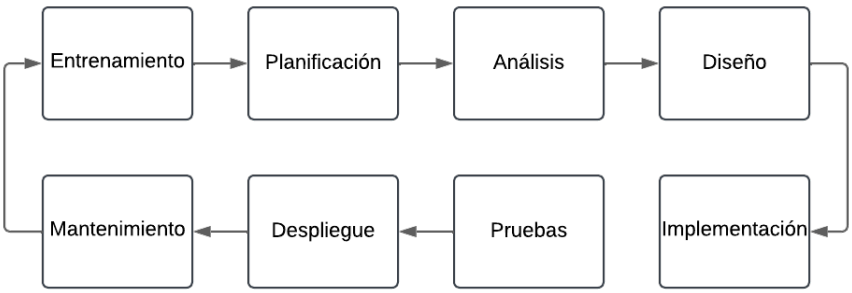

Figura 2.9: Ciclo de Vida del Desarrollo de Software

2.5.2. Ingeniería de requerimientos

Los requerimientos se entienden como descripciones de lo que el sistema debe realizar o poseer, así como de sus limitaciones. El proceso de identificar, analizar, documentar y verificar estos servicios y restricciones se le conoce como ingeniería de requerimientos (IR). Es común no encontrar una definición formal o que sea de común acuerdo en la industria del software. Es importante definir que a nivel de aplicación existen distintos niveles de descripción, los cuales se pueden categorizar como los requerimientos del usuario y los del sistema. Los primeros constan de enunciados de lenguaje natural con diagramas, que expresan lo que esperan los usuarios sobre el servicio y las restricciones. Por otro lado, los requerimientos del sistema son descripciones muy detalladas sobre los mismos puntos, las cuales se documentan y muchas veces forman parte del contrato entre los clientes y los desarrolladores. Usualmente, los requerimientos se clasifican en dos: funcionales y no funcionales. Los requerimientos funcionales

se describen como especificaciones de los servicios que el sistema debe ofrecer y de la forma en que se debe responder ante situaciones o condiciones específicas. Por su parte, los requerimientos no funcionales se definen como limitaciones aplicadas a los servicios o funciones que el sistema proporciona. Estas pueden incluir restricciones de tiempo, plazos de desarrollo y estándares. El proceso de ingeniería de requerimientos se enfoca en elaborar un documento que especifique los requerimientos de todos los interesados en el desarrollo del sistema. [Sommerville, 2005]

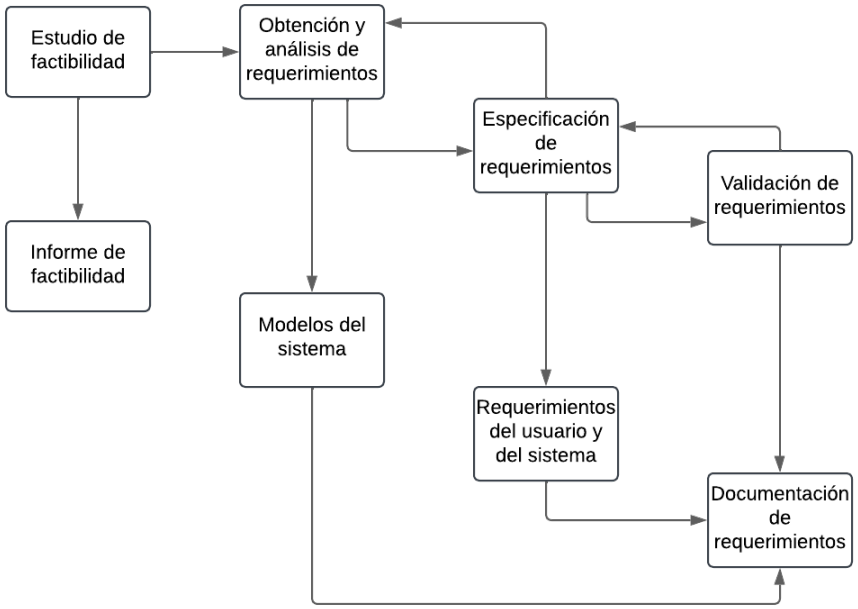

Figura 2.10: Proceso de ingeniería de requerimientos elaborado con material de [Sommerville, 2005]

2.5.3. Diseño de software

El diseño de software abarca un conjunto de conceptos, los cuales guían la creación de un sistema o producto de alto nivel. Mitchell Kapor, en el año 1990, publicó en la revista estadounidense *Dr. Dobbs Journal* un "manifiesto del diseño de software" en el cual mencionaba que el diseño de software no se alejaba del diseño de las personas, como los vitruvios romanos, los cuales contaban con resistencia, funcionalidad y belleza; cualidades que se aplican de igual manera al

software. El ser funcional aplica a que un programa debe ser apropiado para los fines que requiere; resistente, al no tener un error que impida su funcionamiento; y bello, que al momento de usar el programa sea placentero [Kapor, 1990]. El diseño de software es importante ya que permite tener un modelo sobre el sistema o producto que se va a construir, siendo el lugar donde se establece la calidad del software. De manera general, las etapas por las que se divide el diseño de software comienzan con la representación de la arquitectura del sistema o producto; posteriormente se modelan las diversas interfaces que conectan al software con los usuarios finales, con otros sistemas y con sus propios componentes; y, finalmente, se diseñan los componentes que dan paso a la construcción del mismo. [Pressman, 2010].

2.5.4. Arquitectura del sistema

La arquitectura de software se enfoca en entender cómo se debe organizar un sistema y cómo se diseña la estructura del sistema. Es el puente entre la ingeniería de requerimientos y el diseño de software, ya que define los componentes principales de un sistema y la relación entre ellos. La arquitectura de software cuenta con gran importancia, ya que afecta de manera directa el desempeño y la potencia del sistema, al igual que depende de los requerimientos no funcionales. De esta definición, surgen diversos patrones arquitectónicos como una descripción abstracta de compilaciones de buenas prácticas utilizadas anteriormente que han funcionado con éxito, así como la descripción de las fortalezas y debilidades de cada uno de ellos. A continuación, se presenta una tabla con los patrones arquitectónicos más comunes, así como el que se utiliza dentro de este proyecto (ver Cuadro 2.2).

2.5.5. Pruebas de software

Las pruebas de software constituyen un conjunto de actividades esenciales en el proceso de desarrollo, cuyo objetivo principal es evaluar la calidad y robustez del sistema antes de su despliegue (Figura 2.11). Este proceso se articula en varios niveles que permiten identificar y corregir errores de forma temprana, reduciendo costos de mantenimiento y aumentando la satisfacción del usuario final. Incluyen pruebas unitarias, que permiten verificar componentes individuales; pruebas de integración, para validar la interacción entre módulos; pruebas de sistema, para evaluar el sistema completo; y pruebas de aceptación, para asegurar que el software cumple con los requisitos del usuario. En conjunto, estos niveles de pruebas aseguran la detección temprana de errores, lo que resulta

fundamental para evitar costosas rectificaciones y para asegurar un elevado nivel de calidad en el producto final. La implementación de un riguroso proceso de pruebas no solo facilita la corrección de defectos, sino que también contribuye a la documentación y mejora continua del software, aspectos clave para el mantenimiento y evolución futura del sistema [Myers et al., 2011].

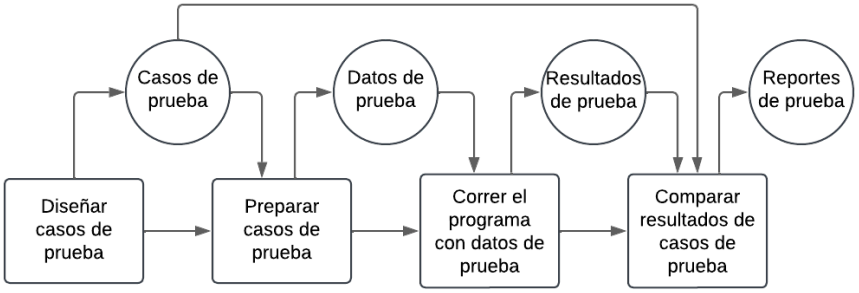

Figura 2.11: Proceso de pruebas de software elaborado con material de [Sommerville, 2005]

Nombre	Uso	Ventajas	Desventajas
MVC (Modelo-Vista-Controlador)	Separa presentación e interacción de los datos del sistema. Está compuesto por tres componentes lógicos que funcionan entre sí. Modelo: Integra la funcionalidad principal y administra los datos. Vista: Se encarga de presentar los datos al usuario, y una misma aplicación puede contar con una o varias vistas. Controlador: Maneja la entrada del usuario.	Permite que los datos cambien de manera independiente. Implica una fácil colaboración. Se recomienda en aplicaciones multi-vista.	Implica código adicional y complejidad que requiere que se tengan conceptos muy claros.
Arquitectura en capas	En este patrón, la estructura del programa se divide en varias capas, cada una de las cuales representa una subtarea y pertenece a un nivel distinto de abstracción.	Permita la sustitución de capas completas en tanto se conserve la interfaz. Facilidad de desarrollo. Cuenta con capacidad de testeo.	En la práctica, es difícil separar limpiamente el sistema en capas, ocasionando que funciones de una capa se encuentren en un nivel que no les corresponde. Presenta un menor rendimiento en comparación con otros patrones, puesto que la solicitud debe pasar individualmente por todas las capas del diseño.
Cliente-servidor	Se fundamenta en el concepto de un servidor encargado de ofrecer y uno o varios clientes, que realizan solicitudes al servidor y reciben las respuestas correspondientes.	La centralización facilita la administración y refuerza la seguridad. Además, la independencia de componentes permite una escalabilidad y mantenimiento sencillos, facilitando migraciones de hardware o software.	El servicio depende del servidor: cualquier caída o congestión implica pérdida total de funcionalidad. Además, la calidad del hardware y software es clave, ya que deficiencias afectan el rendimiento, y al no haber acceso físico al producto, una falla impide continuar el trabajo del cliente.
Tubería y filtro (pipe and filter)	El procesamiento se divide en las tuberías, las cuales transfieren datos entre filtros, los cuales son las unidades encargadas de procesarlos.	Son fáciles de entender y el estilo del flujo del trabajo coincide con la estructura de una gran cantidad de procesos empresariales.	La forma de transferencia de datos debe acordarse entre las transformaciones que se comunican, cada una de estas, debe analizar sus entradas y sintetizar sus salidas al formato acordado. Su arquitectura es lineal, por lo que no se ajusta a aquello que requiera procesamiento bidireccional o paralelo.

Cuadro 2.2: Patrones arquitectónicos más comunes, elaborado con material de [Sommerville, 2005] y [Huet, 2022]

Capítulo 3

Metodología del desarrollo de la aplicación

3.1. Introducción al diseño de la aplicación

El manejo adecuado de la información académica proporciona a la comunidad universitaria beneficios como la mejora del proceso de toma de asistencia en clase, así como la contabilización de la cantidad de alumnos y personal en caso de emergencia. Todo esto con el objetivo de agilizar los procesos y reducir tiempo en procesos cotidianos y de emergencia. De esta manera nace *Check-BUAP*, una aplicación que cuenta con diversas funciones dependiendo del rol en la aplicación:

- Alumnos: Uno de los roles más importantes, ya que al registrarse e iniciar sesión en la aplicación permite generar un código QR (*Quick Response*) el cual escanea el profesor y le permite pasar lista, además de contar con una sección que muestra datos de emergencia en caso de que se requieran.
- Profesores: Para cada materia que imparte el profesor, este cuenta con la posibilidad de escanear el código QR generado por el alumno y de esta manera generar un registro de asistencia en sus clases y visualizarlas en caso de ser necesarios.
- Personal de DASU (Dirección de Apoyo y Seguridad Universitaria): Al personal de DASU le permite visualizar en tiempo real, la cantidad de personas en un edificio, para que en caso de emergencia, puedan agilizar los

procesos de conteo y conocer cuantas personas se encuentran y comparar con los personas durante o después del incidente.

Dentro del flujo de datos de la aplicación, se encuentran datos sensibles como el tipo de sangre, dirección, número de seguridad social, entre otros; los cuales deben resguardarse de manera segura. Por lo que se propone en este rubro implementar un método de encriptación híbrido para asegurar que los datos sensibles se mantengan seguros. De esta manera, surge en esencia *CheckBUAP*.

Importancia del diseño en relación con los objetivos de la aplicación

Teniendo en cuenta los requerimientos y objetivos de la aplicación, se estructuró el sistema enfocado en el usuario que permite un fácil registro de asistencias, así como la protección de la información mediante un sistema de cifrado híbrido. Esta perspectiva permite que la aplicación no solo sea funcional, sino también confiable, promoviendo una adopción amplia dentro de la comunidad universitaria.

3.2. Requerimientos funcionales y no funcionales

Dentro del Capítulo 2, en la Sección 2.5.2, se habla sobre la ingeniería de requerimientos, de la cual se desprenden sus dos categorías: funcionales y no funcionales. El diseño y desarrollo de CheckBUAP se fundamenta en un conjunto de requerimientos cuidadosamente definidos que aseguran el cumplimiento de sus objetivos principales: brindar una solución eficiente para el registro de asistencias, asegurar la protección de los datos personales y brindar una experiencia de usuario satisfactoria. Estos requerimientos, que se desprenden del análisis de necesidades realizado durante las primeras fases del proyecto, han sido categorizados en funcionales y no funcionales para facilitar su implementación y validación posterior.

Requerimientos funcionales

Los requerimientos funcionales de CheckBUAP especifican las acciones concretas que el sistema debe ser capaz de realizar para satisfacer las necesidades de sus usuarios. Para los alumnos, el sistema debe proporcionar un proceso de registro seguro que incluya la validación de datos personales como nombre, correo electrónico institucional, matrícula y contraseña, implementando mecanismos que garanticen la autenticidad de la información proporcionada. Para ello, el sistema emplea el cifrado AES en modo GCM (AES Galois/Counter Mode),

que asegura la integridad y autenticidad de los datos a través de una etiqueta de autenticación. Además, se utiliza cifrado híbrido con RSA para proteger la clave simétrica y se implementan JSON Web Tokens (JWT) firmados digitalmente para validar que las acciones provienen de usuarios autenticados. Una vez registrados, los estudiantes podrán marcar su asistencia a clases y actividades académicas mediante un sistema de códigos QR personalizados, diseñados para prevenir suplantaciones o fraudes en el registro de presencia.
Por parte del personal académico, la aplicación debe ofrecer herramientas completas para la gestión de asistencia, incluyendo la generación automática de códigos QR para cada sesión, la visualización en tiempo real de los alumnos presentes y la generación de reportes detallados que puedan exportarse para su integración con otros sistemas universitarios. Para el personal de la Dirección de Apoyo y Seguridad Universitaria (DASU), el sistema debe proveer un módulo especializado que muestre la ocupación en tiempo real de los diferentes edificios del campus, con capacidad para filtrar la información por edificio, horario y tipo de actividad, facilitando la toma de decisiones durante emergencias o situaciones críticas. Un aspecto fundamental del sistema es su capacidad para manejar información sensible de los usuarios, implementando un esquema de cifrado híbrido que combine los algoritmos RSA y AES en todas las etapas del procesamiento de datos, desde el registro inicial hasta el almacenamiento y las transferencias de información entre componentes del sistema. Este mecanismo de seguridad debe operar de forma transparente para los usuarios finales, sin afectar la usabilidad o el rendimiento de la aplicación.

Requerimientos no funcionales

En este caso, se definieron los siguientes requerimientos no funcionales. Los requerimientos no funcionales de CheckBUAP establecen los criterios de calidad y las condiciones bajo las cuales el sistema debe operar para garantizar su efectividad a largo plazo. En términos de rendimiento, la aplicación debe mantener tiempos de respuesta inferiores a 2 segundos para las operaciones críticas como el registro de asistencia o la generación de reportes, incluso durante períodos de alta demanda cuando puedan concurrir hasta 500 usuarios simultáneos. La arquitectura del sistema debe diseñarse con capacidad de escalamiento horizontal, permitiendo incrementar su capacidad de procesamiento según las necesidades futuras de la universidad. La disponibilidad del sistema es otro aspecto crítico, debiendo garantizar un tiempo de operación continuo de al menos el 80 % mensual, excluyendo únicamente las ventanas de mantenimiento programado que deberán realizarse preferentemente en horarios no académicos.

Para asegurar la accesibilidad, la aplicación debe ser completamente funcional en diversos dispositivos móviles y sistemas operativos, con una interfaz responsiva que se adapte automáticamente a distintos tamaños de pantalla y modos de visualización. La experiencia de usuario se considera prioritaria en el diseño de CheckBUAP, estableciendo como requisito que las tareas principales —como registrar asistencia o consultar reportes— puedan completarse en menos de tres pasos intuitivos. Para ello, se implementan interfaces claras y consistentes que minimicen la curva de aprendizaje, promoviendo la interactividad, al permitir respuestas rápidas a las acciones del usuario; la accesibilidad, facilitando el acceso desde cualquier dispositivo con conexión a Internet; y la actualización continua, asegurando mejoras en tiempo real sin necesidad de instalaciones por parte del usuario [Martinez, 2025]. En cuanto a seguridad y privacidad, el sistema debe cumplir estrictamente con los avisos de privacidad establecidos por la universidad, implementando medidas adicionales como el cifrado de extremo a extremo para la información sensible y mecanismos de autenticación robustos que prevengan accesos no autorizados. Asimismo, en caso de que la aplicación sea utilizada dentro de la universidad, la información registrada se resguardará conforme a los lineamientos institucionales vigentes en materia de protección de datos personales.

3.3. Arquitectura de la aplicación

Arquitectura general

La arquitectura de la aplicación consta de tres campos, de los cuales se hablará con detalle más adelante en las siguientes secciones: frontend, backend y base de datos. En el frontend de la aplicación se compone de la interfaz gráfica que es utilizada por los usuarios; dependiendo del usuario, es el tipo de vista que se muestra para ellos. Se caracteriza por seguir los colores universitarios y contar con un diseño sencillo e intuitivo que permite a los usuarios no requerir muchos recursos al no ser una plataforma compleja. El backend, por otro lado, es aquel encargado de unir el funcionamiento de la aplicación, ya que, además de contar con todo el esqueleto de la aplicación, contiene el núcleo de esta tesis: el sistema híbrido de criptografía; al igual que contiene la conexión a la base de datos, el tercer componente encargado de reservar la información de los usuarios, así como algunos componentes esenciales para el encriptado de la información sensible.

Tecnologías utilizadas

Para el desarrollo de la aplicación, se emplearon herramientas que hacen uso del lenguaje de programación JavaScript debido a su versatilidad tanto en el desarrollo del frontend como del backend, lo que permite un flujo de trabajo unificado entre cliente y servidor, reduciendo la complejidad del mantenimiento y mejorando la eficiencia del desarrollo [Mozilla Developer Network Contributors, 2025]. El frontend se desarrolló con React, una biblioteca de JavaScript ampliamente utilizada para crear interfaces de usuario interactivas y reactivas, junto con HTML5 y CSS3, estándares que permiten la creación de aplicaciones web responsivas y accesibles desde múltiples dispositivos [Meta Platforms, Inc., 2025]. Node.js fue seleccionado por su modelo asincrónico y basado en eventos, el cual permite manejar múltiples solicitudes simultáneamente con bajo consumo de recursos. Express.js se utilizó por su simplicidad y eficiencia para construir APIs RESTful, facilitando la comunicación entre el cliente y el servidor. Para la implementación del cifrado, se utilizaron las bibliotecas crypto (incluida nativamente en Node.js) y node-rsa, las cuales permiten aplicar algoritmos de cifrado simétrico (AES) y asimétrico (RSA). Estas bibliotecas proporcionan funcionalidades criptográficas confiables para proteger los datos sensibles conforme a buenas prácticas de seguridad [OpenJS Foundation, 2025]. PostgreSQL fue elegido como sistema gestor de bases de datos por su robustez, cumplimiento con estándares SQL y soporte avanzado para transacciones. Su estructura permite el manejo eficiente de datos cifrados sin comprometer la integridad de las operaciones [The PostgreSQL Global Development Group, 2025].

Flujo de datos

En el diagrama de flujo de datos 3.1 se muestra cómo los usuarios interactúan con el sistema a través del frontend, enviando información que es procesada por el backend. Los datos ingresados son validados, cifrados y almacenados en una base de datos PostgreSQL. Posteriormente, los administradores pueden solicitar reportes que son extraídos, descifrados y devueltos al frontend. Este flujo asegura la protección de la información mediante el uso de cifrado híbrido y garantiza una experiencia segura y eficaz para los usuarios.

Seguridad de la arquitectura

Siendo la columna de este trabajo de tesis la seguridad en la aplicación, se cuenta con diversas implementaciones que aseguran una protección de la infor-

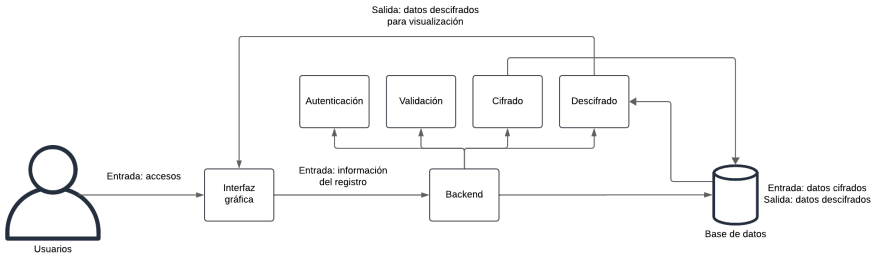

Figura 3.1: Diagrama del flujo de datos

mación de los usuarios. El principal componente se centra en el cifrado híbrido para proteger la información en tránsito y en reposo, así como en la autenticación basada en credenciales y controles de roles basados en el esquema RBAC (Role Based Access Control) ya que, dependiendo del tipo de rol asignado, se cuenta con los accesos correspondientes a sus funciones. Además, se cuentan con medidas implementadas contra ataques como SQL Injection y Cross-Site Scripting (XSS) que forman parte de las medidas contra ataques específicos que son recurrentes ante la falta de configuración en el desarrollo de la aplicación, asegurando la integridad de la información. Además de ello, se hablará en el trabajo conjunto de la implementación de políticas de respaldo y buenas prácticas como medidas de buenas prácticas para el uso de la aplicación a nivel productivo, es decir, en caso de uso dentro de la universidad.

3.4. Diseño del frontend

El frontend de la aplicación es la interfaz con la que interactúan los usuarios, proporcionando un medio intuitivo y eficiente para registrar asistencias, consultar eventos y acceder a información relevante. Su diseño se enfoca en la usabilidad, accesibilidad y seguridad de la información, garantizando una experiencia fluida y coherente en distintos dispositivos. Para su desarrollo se utilizaron tecnologías modernas que permiten construir interfaces reactivas y modulares, optimizando el rendimiento y la experiencia del usuario. Las tecnologías utilizadas en el frontend incluyen:

- **HTML5**, **CSS3** y **JavaScript** para la estructura, estilos y funcionalidad base del sitio web.
- **React.js** como biblioteca principal para la creación de componentes dinámicos y reutilizables, facilitando la gestión del estado y la navegación.

- **Bootstrap** y **SASS** para el diseño responsivo, la personalización visual y la adaptación eficiente a dispositivos móviles.

Principios de diseño

Dentro de la interfaz de usuario, se prioriza la simplicidad para facilitar la navegación del usuario, al igual que cuenta con diseño responsivo, lo que la hace una aplicación compatible con dispositivos móviles, tabletas y computadoras de escritorio. Un diseño importante es el feedback visual, ya que se generan alertas y notificaciones para indicar el éxito o error en las acciones del usuario. Por último, la aplicación cuenta con accesibilidad, ya que cumple con las pautas de accesibilidad WCAG 2.1, las cuales indican que el contenido web debe ser: perceptible, es decir, que el contenido puede verse de manera fácil; operable, que tenga acceso a las funciones por el teclado y que conceda a los usuarios el tiempo necesario para leer el contenido; comprensible, que pueda entenderse y evite que los usuarios cometan errores; y, por último, robusto, que sea compatible con herramientas tecnológicas actuales y futuras. [Web Accessibility Initiative (WAI), 2025]

3.5. Diseño del backend

El backend de la aplicación es el responsable de procesar la lógica del sistema, administrar la comunicación con la base de datos y asegurar la protección de los datos sensibles. Está diseñado bajo una arquitectura modular que facilita su mantenimiento, escalabilidad y reutilización de componentes. Entre las tecnologías utilizadas en el backend se encuentran:

- **Node.js** junto con **Express.js** para la creación y gestión de rutas, middleware y controladores de la API REST.
- **PostgreSQL** como sistema gestor de base de datos relacional, robusto y adecuado para el manejo de datos estructurados.
- **JWT (JSON Web Tokens)** para la autenticación de usuarios, permitiendo el acceso seguro a los recursos protegidos del sistema.
- **Bcrypt** para el almacenamiento seguro de contraseñas mediante hashing, lo que impide su recuperación en texto plano.
- Librerías criptográficas como **Crypto** y **Node-RSA** para la implementación de un esquema de cifrado híbrido basado en AES (simétrico) y RSA (asimétrico), el cual garantiza la confidencialidad e integridad de los datos sensibles.

Entre las principales funcionalidades implementadas en el backend se incluyen la gestión de usuarios, que abarca el registro, validación, autenticación y autorización con control de acceso basado en roles (RBAC). También se encarga del registro de asistencias, generando códigos QR para los estudiantes y validándolos a través de los dispositivos del personal docente. Además, el sistema contempla el manejo centralizado de errores y la implementación de medidas básicas de seguridad para prevenir ataques como inyecciones SQL, XSS (Cross-Site Scripting) y fuerza bruta en autenticación. Finalmente, el backend es responsable del proceso de cifrado y almacenamiento de datos, utilizando el esquema híbrido que garantiza la confidencialidad de la información crítica registrada en la plataforma.

3.6. Cifrado híbrido

La aplicación CheckBUAP protege la información sensible de los usuarios mediante un esquema de cifrado híbrido que combina los algoritmos AES (Advanced Encryption Standard) y RSA (Rivest-Shamir-Adleman). Esta estrategia busca aprovechar la eficiencia del cifrado simétrico (AES) y la seguridad del cifrado asimétrico (RSA), garantizando así integridad, la confidencialidad y la autenticidad de los datos almacenados.

1. Generación de la clave simétrica AES

Para cada usuario o dato cifrado, se genera una clave AES de 256 bits utilizando la función `crypto.randomBytes()`:

```
Funcion generateAESKey():
Retornar claveAleatoria de 256 bits
```

Esta clave se utiliza exclusivamente para cifrar los datos sensibles (como nombre, matrícula, contacto, etc.) y no se almacena en texto plano.

2. Cifrado de datos con AES-GCM

La información del usuario es cifrada utilizando el modo AES-GCM (Galois/Counter Mode), el cual permite cifrar y generar automáticamente un tag de autenticación para verificar la integridad del contenido cifrado:

```
Funcion encriptarDatos(datos, claveAES):
    IV ← generarIVAleatorio()
    datosCifrados ← cifrarConAESGCM(datos, claveAES, IV)
```

```
    authTag ← obtenerAuthTag()
Retornar (datosCifrados, IV, authTag)
```

Este proceso genera tres elementos críticos:

- **encriptarDatos**: contenido cifrado del dato original.
- **iv**: vector de inicialización, necesario para descifrar.
- **authTag**: etiqueta que garantiza que los datos no fueron alterados.

3. Protección de la clave AES con RSA

Dado que la clave AES es esencial para descifrar la información, esta se cifra utilizando el algoritmo RSA y la clave pública del sistema. De este modo, solo el backend, que posee la clave privada, puede recuperarla:

```
Funcion encriptarLlaveAES(claveAES, clavePublicaRSA):
    claveAESCifrada ← cifrarConRSA(claveAES, clavePublicaRSA)
Retornar claveAESCifrada
```

4. Estructura del objeto cifrado

El resultado del proceso de cifrado es un objeto que contiene todos los elementos necesarios para el descifrado, pero sin exponer información sensible en texto claro:

```
ObjetoCifrado ← { datosCifrados, IV, authTag, claveAESCifrada }
```

Estos valores se almacenan en la base de datos como parte de los registros del usuario, permitiendo su posterior recuperación y descifrado.

5. Descifrado seguro y validación de integridad

Para recuperar la información, el backend primero descifra la clave AES utilizando la clave privada RSA. Luego, se configura el descifrador con el mismo vector de inicialización (IV) y se comprueba la autenticidad de los datos mediante la etiqueta de autenticación (authTag). Si alguno de estos componentes no coincide con los originales, el descifrado falla automáticamente.

```
Funcion desencriptarDatos(objetoCifrado, clavePrivadaRSA):
   claveAES ← descifrarConRSA(objetoCifrado.claveAESCifrada,
              clavePrivadaRSA)
```

```
        datos ← descifrarConAESGCM( objetoCifrado.datosCifrados,
                claveAES, objetoCifrado.IV, objetoCifrado.authTag)
    Retornar datos
```

6. Beneficios del esquema híbrido

El uso combinado de AES-GCM y RSA proporciona las siguientes garantías de seguridad:

- **Confidencialidad:** los datos no pueden ser leídos sin la clave AES, que a su vez está protegida con RSA.
- **Autenticidad e integridad:** el authTag generado por AES-GCM impide el uso de datos manipulados.
- **Separación de responsabilidades:** las claves nunca se transmiten ni almacenan en texto plano.
- **Resistencia a ataques:** la clave AES es distinta por usuario o dato, y el uso de IV aleatorios evita patrones de cifrado.

Este esquema cumple con los principios fundamentales de la criptografía aplicada, reforzando la seguridad del sistema en cada punto donde se maneja información sensible.

3.7. Base de datos

El sistema CheckBUAP utiliza PostgreSQL como motor de base de datos relacional, seleccionado por su robustez, soporte a transacciones seguras y su capacidad para integrarse eficazmente con el cifrado de datos en entornos sensibles. La base de datos almacena información fundamental para el funcionamiento del sistema, como los datos de alumnos, docentes, horarios académicos, registros de asistencia y datos de autenticación de usuarios. Su estructura ha sido diseñada bajo principios de normalización para mantener la consistencia referencial entre las distintas entidades, permitiendo establecer relaciones claras entre alumnos, clases y horarios, lo cual es clave para generar y validar los códigos QR utilizados en el pase de lista. La protección de la información se ha reforzado mediante el uso de un esquema de cifrado híbrido implementado desde el backend. Aunque el cifrado y descifrado se realizan fuera de la base de datos, PostgreSQL almacena los datos cifrados (incluyendo el iv, el authTag y la clave AES cifrada con RSA) de manera organizada y separada por campo, permitiendo preservar la confidencialidad y autenticidad de la información incluso en

el almacenamiento. Además, se manejan niveles de acceso diferenciados según el tipo de usuario (alumno, docente o administrador), lo que permite restringir adecuadamente la visualización y edición de datos dentro del sistema.

3.8. Vistas de la aplicación

El diseño de la interfaz de usuario en CheckBUAP ha sido desarrollado bajo principios de usabilidad, accesibilidad y seguridad, siguiendo un enfoque centrado en las necesidades específicas de cada tipo de usuario (alumnos, profesores y personal administrativo). La aplicación implementa un flujo de navegación intuitivo que guía al usuario a través de cada funcionalidad clave, desde el primer acceso hasta las operaciones diarias. A continuación, se describen las vistas principales que conforman la experiencia de usuario.

Autenticación y registro

La aplicación inicia con un sistema de acceso seguro, véase en la figura 3.2 que valida las credenciales mediante autenticación cifrada. Para nuevos usuarios, se ha diseñado un formulario de registro multipaso, en las figuras 3.3 y 3.4, que recopila información esencial como matrícula, datos personales y credenciales médicas de emergencia, aplicando validaciones en tiempo real y notificaciones claras para prevenir errores. Este proceso cumple con los estándares de protección de datos al encriptar toda la información sensible antes de su almacenamiento.

Interfaz principal y gestión de datos de los alumnos

Tras el acceso, los alumnos encuentran su panel principal, véase la figura 3.5, con un código QR único, generado dinámicamente mediante algoritmos criptográficos que evitan su replicación o suplantación. La vista incluye opciones para mostrar el código QR para registro de asistencia y editar información personal, véase la figura 3.6.

Vista especializadas para roles académicos y de seguridad

El sistema CheckBUAP ha sido diseñado con interfaces diferenciadas para cada tipo de usuario, optimizando las funcionalidades según sus responsabilidades específicas. A continuación, se presentan las vistas clave para el personal académico y de seguridad. Para el caso del panel de control DASU, la Figura

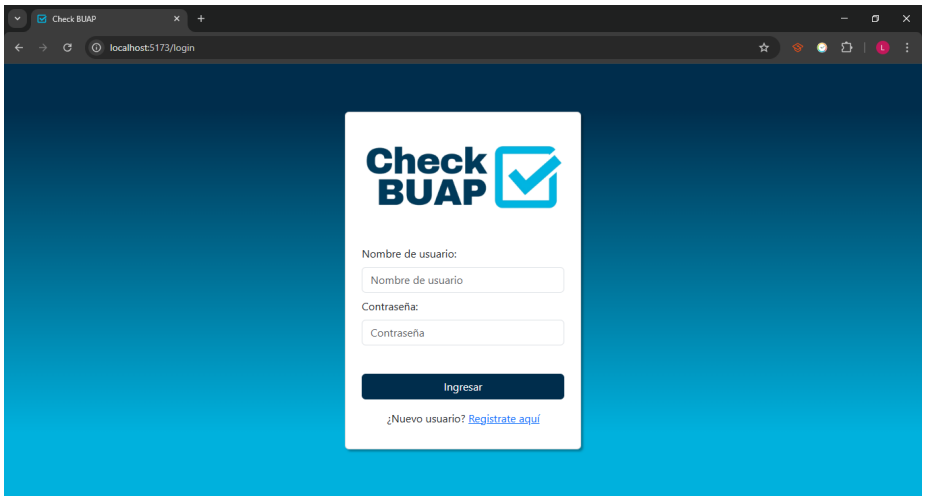

Figura 3.2: Login inicial; es la primera pantalla que se muestra al acceder a la aplicación.

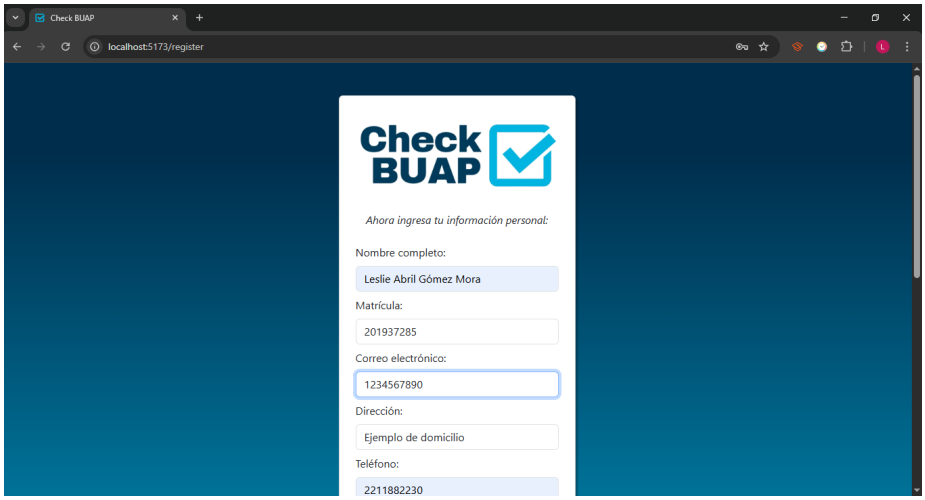

Figura 3.3: Primera parte del registro de datos, en caso de no contar con un usuario.

3.7 muestra el *dashboard* especializado para el personal de seguridad, que proporciona información crítica en tiempo real sobre la ocupación de los edificios universitarios. Como se observa en las Figuras 3.8 y 3.9 los profesores cuentan con una plataforma de gestión de asistencia integrada que permite la generación automática de sesiones QR para cada clase y la visualización de estadísticas de

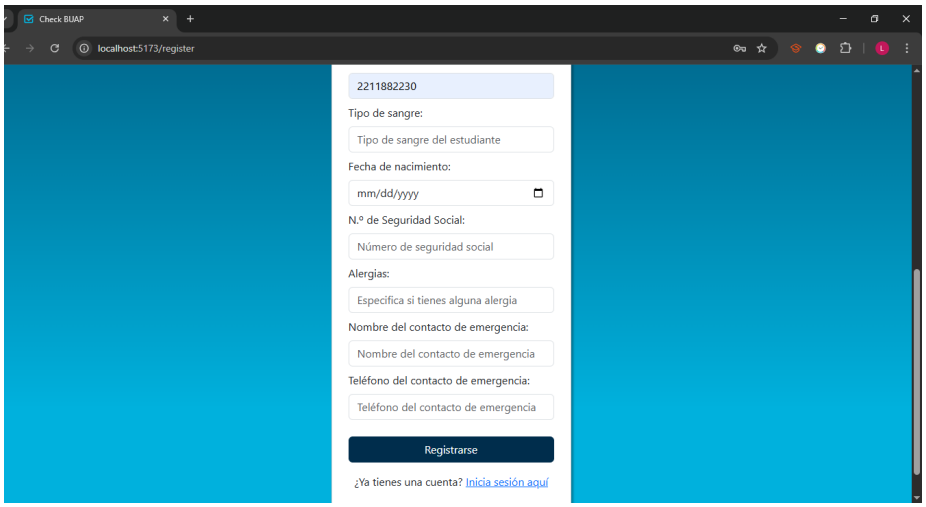

Figura 3.4: Segunda parte del registro de datos, en caso de no contar con un usuario.

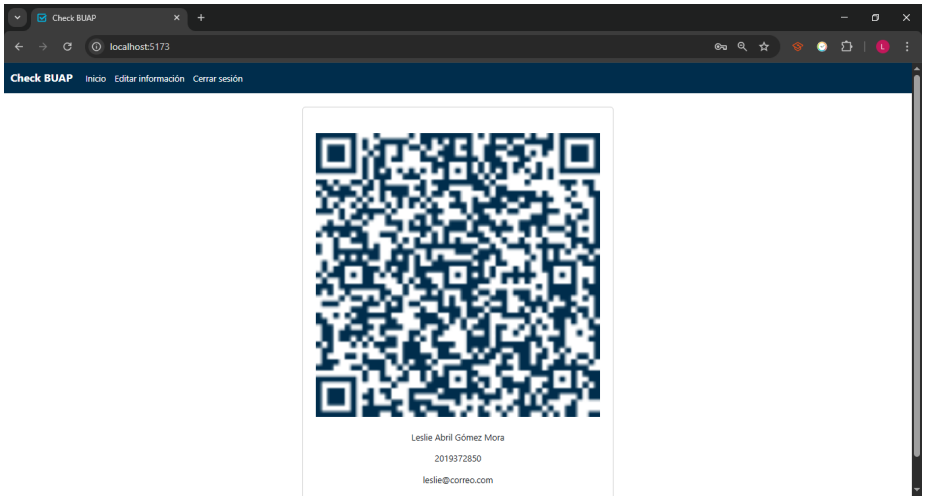

Figura 3.5: Después del acceso se muestra inmediatamente un el código QR del estudiante.

asistencia histórica y en tiempo real.

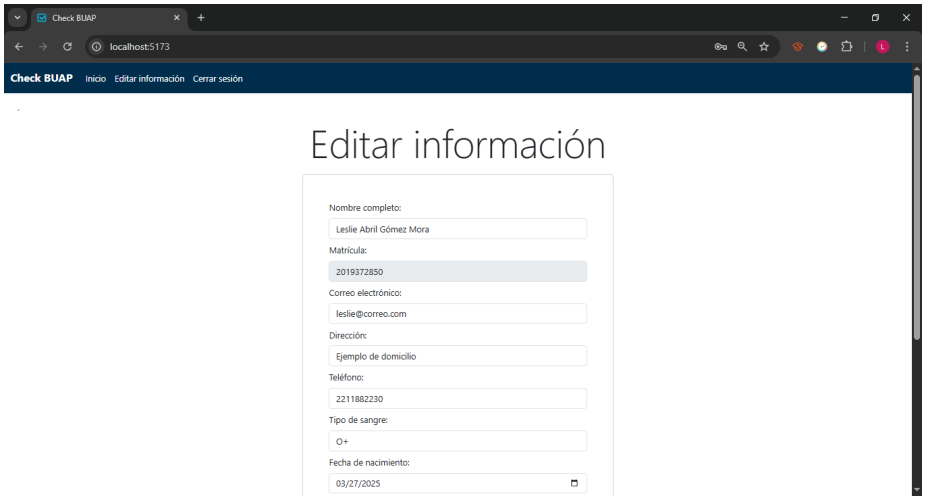

Figura 3.6: En caso de modificación de datos, se permite editarlos.

Figura 3.7: Página de inicio del personal de DASU, permite la visualización de las personas en el edificio.

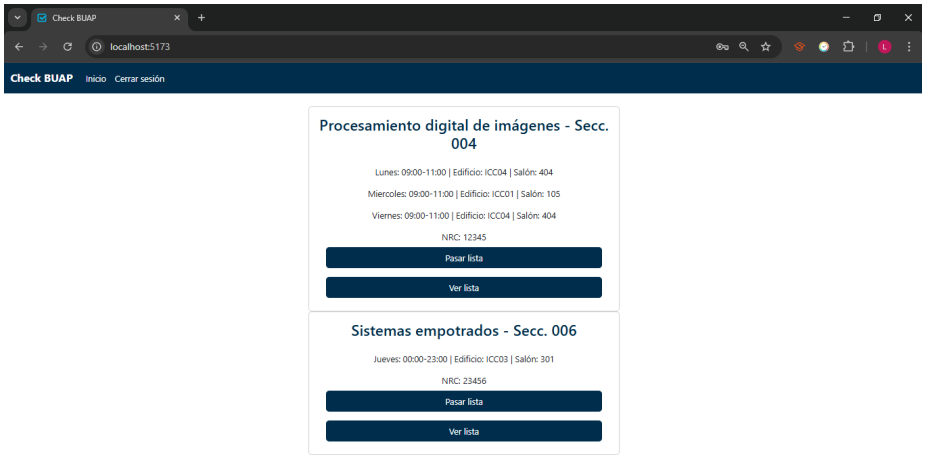

Figura 3.8: Página de inicio del personal académico, permite las acciones de pase de lista y la visualización de la asistencia de esa clase.

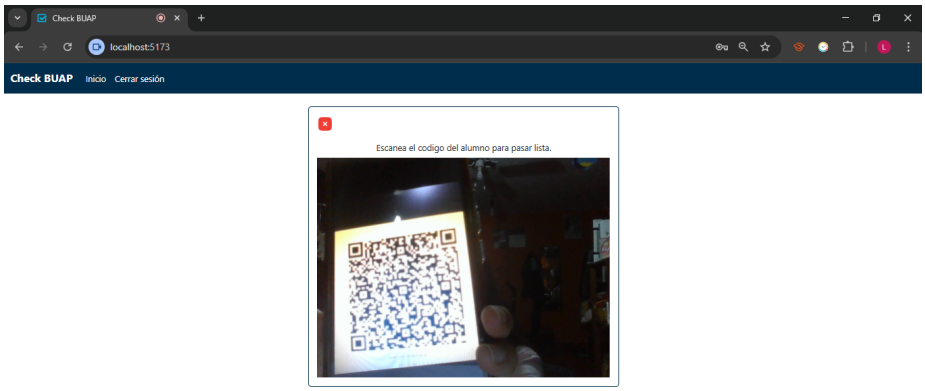

Figura 3.9: Escaneo del código QR del alumno

Capítulo 4

Pruebas y resultados

4.1. Evaluación de criptografía segura en la aplicación

El principal objetivo de *CheckBUAP* es ofrecer una plataforma de gestión segura para estudiantes universitarios, integrando procesos de autenticación y almacenamiento de información sensible como nombres, direcciones, teléfonos, contactos de emergencia y números de seguridad social. Debido a la sensibilidad de los datos tratados y al entorno académico en el que opera, la seguridad se convierte en un eje transversal para asegurar la confidencialidad, integridad y disponibilidad de la información. Es aquí donde se une con el concepto de seguridad *PII* (*Personal Identifiable Information*, por sus siglas en inglés), el cual se define como cualquier información conectada a un individuo en específico que puede ser utilizada para encubrir o robar la identidad de ese individuo. Este concepto ha tomado relevancia en las últimas décadas; es casi indispensable el uso de las tecnologías de la información tanto en la vida personal como en el trabajo, lo que ha permitido el incremento del uso de la información personal que se comparte diariamente [IBM, 2025]. Debido a esto, se decidió realizar una evaluación específica alineada con el estándar OWASP Top 10:2021, un marco de referencia ampliamente adoptado para identificar y priorizar amenazas en aplicaciones web; al igual que se toma de referencia la legislación actual del país, en este contexto la Ley Federal de Protección de Datos Personales en Posesión de Particulares (LFPDPPP). A pesar de contar con diversas clasificaciones, este trabajo se enfoca en particular en las categorías del OWASP A02:2021 "Fallas Criptográficas" y A07:2021 "Fallas de Identificación y Autenticación". La

primera categoría pasó a ocupar el segundo lugar del ranking de riesgos en comparación con su edición anterior, 2017. Esta categoría, anteriormente conocida como "Exposición de Datos Sensibles", ahora se orienta explícitamente a errores de implementación criptográfica y configuraciones débiles que derivan en pérdida de confidencialidad o integridad [OWASP, 2021]. Por otro lado, la categoría A07:2021 "Fallas de Identificación y Autenticación" se enfoca en mitigar ataques como fuerza bruta, sesiones inseguras o validaciones de credenciales inadecuadas. Estas categorías agrupan múltiples debilidades, incluyendo: CWE-259 (uso de contraseñas en el código fuente), CWE-327 (uso de algoritmos criptográficos inseguros), CWE-331 (entropía insuficiente en la generación de claves o vectores de inicialización), CWE-307 (número de intentos de autenticación no limitado), entre otras. Estas debilidades son listas del sitio oficial de OWASP, las cuales son denominadas *CWE* por sus siglas en inglés, *Common Weakness Enumeration*, las cuales son una lista de tipos de debilidades de software y hardware que pueden implicar vulnerabilidades si no se corrigen. Este sistema es mantenido por la compañía de Mitre, la cual es conocida por ofrecer otros marcos de referencia como el MITRE ATT&CK [Mitre,].

4.1.1. Objetivo de la evaluación

El propósito de esta evaluación es verificar que la aplicación *CheckBUAP* cumple con las recomendaciones del OWASP Top 10:2021 respecto a la protección criptográfica de datos sensibles y demostrar que:

- La información personal no se expone en texto plano, ni en tránsito ni en reposo.
- Se emplean algoritmos criptográficos modernos y autenticados como AES-GCM.
- Las claves y vectores de inicialización se generan de manera aleatoria y no se reutilizan.
- La protección criptográfica es efectiva incluso si otras capas del sistema, como la validación de entrada, son vulneradas.

4.1.2. Metodología de prueba

El procedimiento adoptado para esta evaluación consistió en la simulación de escenarios reales de ataque, replicando los ataques más comunes presentados por el marco de referencia OWASP Top 10:2021.

Verificación de cifrado autenticado

La aplicación *CheckBUAP* implementa AES-GCM para garantizar la confidencialidad así como la integridad de los datos. Para validar su efectividad, se configuró un endpoint temporal GET en Express.js denominado /test-sqlmap, el cual es intencionadamente vulnerable a una inyección SQL (SQLi), lo que sucede cuando un atacante introduce o "inyecta" código SQL malicioso en una consulta, permitiendo así manipular la base de datos con el objetivo de obtener información sensible o la modificación/eliminación de datos. [Ayllapan, 2025]. Un endpoint se define como una URL (Uniform Resource Locator, por sus siglas en inglés) que se usa para acceder a diferentes recursos en la aplicación; en este caso, permite consultar información de estudiantes con el anterior propósito indicado [Foster, 2023].

```
app.get('/test-sqlmap', async (req, res) => {
  const studentId = req.query.id;
  const result = await pool.query
  (`SELECT * FROM students WHERE student_id = '${studentId}'`);
  res.json(result.rows);
});
```

Este endpoint extrae el parámetro id desde la URL (req.query.id) y lo inserta directamente en una consulta SQL mediante interpolación de cadenas, sin ningún mecanismo de sanitización o validación. Al ejecutar pool.query(...), el valor recibido se incorpora directamente en la cláusula WHERE de la consulta `SELECT * FROM students WHERE student_id = '${studentId}'`, lo que habilita un escenario deliberadamente vulnerable. La finalidad de esta prueba consiste en evidenciar que, aunque exista una vulnerabilidad lógica explotable, los datos almacenados se encuentran protegidos mediante cifrado fuerte (AES-GCM), lo cual está alineado con los principios de OWASP. Los siguientes comandos ejecutan *SQLMap*, la cual es una herramienta utilizada para explotar vulnerabilidades por medio de inyección SQL.

```
sqlmap -u "https://localhost:4000/test-sqlmap?id=20351234"
--batch --force-ssl
sqlmap -u "https://localhost:4000/test-sqlmap?id=20351234"
--batch --force-ssl -D public -T students --dump
```

El primer parámetro de este comando es "-u https://url" el cual indica la URL vulnerable a atacar, la parte de "id" indica la matrícula ficticia del estudiante para vulnerar sus datos en este ejercicio. "–batch" ejecuta el proceso en

modo automático, sin pedirte confirmaciones. Usa las opciones por defecto en las preguntas; y por último,"–force-ssl" fuerza a *SQLMap* a usar HTTPS (en caso de que no lo detecte bien automáticamente), esta es útil cuando el servidor requiere SSL/TLS.
Al realizar la prueba, *SQLMap* detectó múltiples técnicas de inyección y logró extraer datos. Sin embargo, los campos sensibles estaban cifrados, de una manera similar a la mostrada en el siguiente fragmento de código.

```
{
  "address": "8bf2a3d912e4c9a7d...",
  "phone_number": "1c2a3d7fd882e4c1aefc8c...",
  "social_security_number": "5dce40a0f9be6a917b..."
}
```

Esto valida el uso correcto de AES-GCM, considerado seguro por NIST (SP 800-38D). Se observó cifrado por campo con IVs únicos, etiquetas de autenticación y claves AES cifradas con RSA. Además, se implementó HTTPS y encabezados de seguridad HTTP con Helmet.

Rechazo de datos alterados en caso de alteración

La siguiente evaluación consiste en detectar y rechazar intentos de descifrado de datos que han sido modificados maliciosamente para alterar la integridad de los datos, y con esta evaluación se comprueba que protege este rubro, además de la confidencialidad. Se accedió directamente a la base de datos PostgreSQL y se modificó la información de uno de los campos cifrados del alumno de prueba. En este caso se alteró uno de los caracteres del cifrado del campo de dirección; esta modificación no mantenía la estructura original del texto cifrado, por lo tanto, rompía su validez criptográfica. Para comprobar esto, se intentó consultar al alumno mediante el inicio de sesión, pero el algoritmo detectó esta modificación, lo que generó el siguiente mensaje:

```
{
 Error en decryptData (GCM):
 Unsupported state or unable to authenticate data

}
```

Este error fue generado en el backend, durante el proceso de descifrado final, y ocurre cuando el tag de autenticación no puede verificar la autenticidad e integridad del texto cifrado modificado; por lo tanto, el sistema rechaza los

datos alterados y no expone ningún contenido sensible, lo que confirma que el sistema es efectivo.

4.1.3. Análisis de cumplimiento OWASP A02:2021

Criterio de seguridad	Cumple	Observación
Cifrado en tránsito mediante TLS	Sí	HTTPS activo con certificado autofirmado
Cifrado en reposo (AES-GCM)	Sí	Cifrado por campo con IVs y auth tags
Algoritmos modernos y autenticados	Sí	AES-GCM y RSA, evitando ECB, SHA1, MD5
Gestión de claves segura	Sí	Claves AES cifradas con RSA
IVs seguros y no reutilizables	Sí	CSPRNG (crypto.randomBytes)
Encabezados HTTP de seguridad	Sí	Helmet con CSP, HSTS, sin caché
Hash de contraseñas seguro	Parcial	Se sugiere uso de bcrypt o Argon2

Cuadro 4.1: Cumplimiento de OWASP A02:2021 en CheckBUAP

Como complemento a la explotación del endpoint vulnerable, se realizaron verificaciones técnicas específicas para validar el cumplimiento de los puntos clave definidos por OWASP A02:2021. En primer lugar, se constató que la comunicación entre cliente y servidor se lleva a cabo a través de HTTPS, lo cual fue comprobado tanto en el navegador como mediante herramientas como Wireshark y curl. En cuanto al cifrado en reposo, la extracción de datos a través de *SQLMap* reveló que los campos sensibles estaban cifrados, lo que evidencia el uso efectivo de AES-GCM, tal como se definió en los módulos de cifrado del backend (encryptionUtils.js). Asimismo, se verificó que las claves AES utilizadas no solo se generan de manera aleatoria, sino que además son cifradas con RSA antes de ser almacenadas en la base de datos. Los vectores de inicialización (IVs) se generan utilizando crypto.randomBytes(12), garantizando así unicidad y entropía adecuada por campo. Se validó que cada campo cifrado guarda su propio IV y etiqueta de autenticación, y que no hay reutilización. Por otra parte, se comprobó la presencia de encabezados de seguridad HTTP en las respuestas del servidor, habilitados mediante el middleware Helmet. Finalmente, respecto al almacenamiento de contraseñas, se identificó que aún no se aplica una función de hash robusta como bcrypt, por lo que se recomienda implementar esta medida para cumplir completamente con los criterios de OWASP en ese rubro. Estas verificaciones permiten afirmar que CheckBUAP implementa mecanismos de protección criptográfica sólidos y que cumple, en gran parte, con los controles sugeridos por OWASP en materia de protección de datos sensibles.

4.1.4. Conclusiones de la prueba y recomendaciones

La evaluación realizada confirma que CheckBUAP cumple con los principios de seguridad establecidos por OWASP A02:2021. Aún ante un escenario de compromiso del sistema mediante inyección SQL, los datos sensibles extraídos se mantienen ilegibles gracias a una implementación robusta de criptografía en reposo con AES-GCM y gestión segura de claves. Esto confirma la capacidad de la aplicación para salvaguardar la confidencialidad de la información estudiantil y refuerza su conformidad con los principios de diseño seguro promovidos en OWASP Top 10. Tal como lo enfatiza OWASP, el enfoque en causas raíz —como el uso correcto de la criptografía— es fundamental para prevenir exposiciones masivas de datos, incluso cuando existen vulnerabilidades técnicas presentes en el sistema. Esta práctica, combinada con controles complementarios como TLS, 'headers' de seguridad y análisis independiente, posiciona a CheckBUAP como una plataforma que es funcional y segura para los usuarios, cumpliendo con marcos de referencia como el OWASP y la LFPDPPP en México.

Capítulo 5

Conclusiones y trabajo a futuro

El desarrollo del Sistema de pase de lista con cifrado híbrido, para la comunidad universitaria constituye una contribución importante en la búsqueda de soluciones tecnológicas que prioricen la protección de los datos personales sin comprometer la eficiencia operativa. A lo largo del proceso de diseño e implementación, se abordaron problemáticas clave relacionadas con la gestión de asistencias en entornos educativos, enfocándose especialmente en garantizar la seguridad, confidencialidad y accesibilidad de la información.

Uno de los principales logros del sistema fue la integración de un mecanismo de cifrado híbrido que combina los algoritmos RSA y AES, permitiendo proteger la información tanto en tránsito como en reposo. Este enfoque técnico, acompañado de una arquitectura modular y escalable, aseguró una base sólida para el crecimiento futuro del sistema. Asimismo, se priorizó una experiencia de usuario clara e intuitiva, de modo que tanto estudiantes como administradores pudieran interactuar de forma sencilla con la plataforma. Las pruebas funcionales realizadas evidenciaron que el sistema cumple adecuadamente con los objetivos planteados, permitiendo registrar asistencias y consultar reportes de manera ágil, segura y eficiente.

Sin embargo, este proyecto abre la puerta a nuevas posibilidades de mejora e integración. Considerando que la Benemérita Universidad Autónoma de Puebla ha desarrollado recientemente una aplicación institucional orientada a brindar servicios como consulta de matrícula, horarios, noticias y calificaciones, resulta pertinente explorar la posibilidad de incorporar un módulo de pase de lista den-

tro de dicha plataforma. Esta integración permitiría unificar diversos servicios académicos en una sola aplicación oficial, mejorando la experiencia del usuario, reduciendo la fragmentación tecnológica y promoviendo una administración académica más eficiente. Además, dotar a esa aplicación de un componente seguro de control de asistencias, basado en el modelo criptográfico propuesto en esta tesis, reforzaría las políticas institucionales de protección de datos personales.

Adicionalmente, el sistema podría ampliarse mediante la incorporación de mecanismos avanzados de autenticación, como firma digital o verificación biométrica, con el fin de fortalecer la validación de identidad. También sería útil desarrollar funcionalidades orientadas al análisis de datos, tales como tableros con métricas en tiempo real que permitan a las autoridades académicas tomar decisiones de manera más informada. De igual manera, una modalidad offline permitiría registrar asistencias en entornos con conectividad limitada, con sincronización automática posterior.

Finalmente, para asegurar la inclusión de todos los perfiles de usuarios, sería recomendable realizar una evaluación más amplia de usabilidad y accesibilidad, así como alinear el sistema con normativas nacionales e internacionales en materia de protección de datos personales. En conjunto, estos pasos contribuirían a robustecer una solución que ya ha evidenciado su viabilidad técnica y su impacto potencial dentro del entorno universitario.

En resumen, este proyecto ha evidenciado que es posible desarrollar una herramienta tecnológica segura, funcional y centrada en el usuario que responde a una necesidad concreta del contexto académico. Su diseño deja abierta la posibilidad de evolucionar hacia un sistema más integral y conectado con el ecosistema digital institucional, contribuyendo así al fortalecimiento de la transformación digital de la universidad.

Bibliografía

[OpenJS Foundation, 2025] OpenJS Foundation (2025).

[The PostgreSQL Global Development Group, 2025] The PostgreSQL Global Development Group (2025).

[Mozilla Developer Network Contributors, 2025] Mozilla Developer Network Contributors (2025). Javascript guide.

[Meta Platforms, Inc., 2025] Meta Platforms, Inc. (2025). React legacy documentation.

[Ayllapan, 2025] Ayllapan, W. (2025). *Dominando PHP y MySQL: de principiante a experto en desarrollo web*. Walter Ulises Ayllapan.

[Busca and Marrón, 2010] Busca, P. and Marrón, R. (2010). La informatización en urgencias y emergencias. *Anales del Sistema Sanitario de Navarra*, 33(Supl. 1):69–76. Recuperado el 13 de marzo de 2025.

[Candel, 2020] Candel, J. (2020). *Desarrollo seguro en ingeniería del software.: Aplicaciones seguras con Android, NodeJS, Python y C++*. Marcombo.

[Carracedo and Alix, 1992] Carracedo, C. and Alix, M. (1992). *Análisis de una variable real*. Reverté.

[de Tejada and Manzano, 2023] de Tejada, J. M. D. F. G.-R. and Manzano, L. G. (2023). *Sistemas seguros de acceso y transmisión de datos. IFCT0109*. IC Editorial, España.

[Foster, 2023] Foster, L. (2023). *Hacking APIs - A Comprehensive Guide from Beginner to Intermediate*. Career Kick Start Books, LLC.

[García et al., 2002] García, C., López, J., and Puigjaner, D. (2002). *Matemática discreta*. Prentice práctica. Pearson Educación.

[Huet, 2022] Huet, P. (2022). Arquitectura de software: Qué es y qué tipos existen.

[IBM, 2025] IBM (2025). What is personally identifiable information (pii)?

[ISC2, 2022] ISC2 (2022). *CC Official ISC2 eTextbook*. ISC² - International Information System Security Certification Consortium, Inc., 1st edition. Accessed: [fecha de consulta, opcional].

[Julio Mulero and (EDS.), 2024] Julio Mulero, L. S. and (EDS.), J. M. S. (2024). *Matemáticas infinitas: Curiosidades y hechos matemáticos*. Publicacions Universitat Alacant, España.

[Kapor, 1990] Kapor, M. (1990). A software design manifesto.

[Kendall et al., 2005] Kendall, K., Kendall, J., and Ramos, A. (2005). *Análisis y diseño de sistemas*. Pearson Educación. Pearson Educación.

[Maiorano, 2009] Maiorano, A. (2009). *Criptografía: Técnicas de desarrollo para profesionales*. Alpha Editorial, Colombia.

[Mammeri, 2024] Mammeri, Z. Z. (2024). *Cryptography: Algorithms, Protocols, and Standards for Computer Security*. Wiley, Reino Unido.

[Mancera Corcuera, 2016] Mancera Corcuera, C. (2016). Las intervenciones para abatir el abandono escolar en educación media superior en méxico: diseño, operación y resultados de la estrategia "yo no abandono", desde la mirada de los actores escolares y las autoridades educativas.

[Martinez, 2025] Martinez, J. (2025). *Fundamentos del Diseño Web: Una Guía para Empezar desde Cero*. JuanMartinez.

[Mata et al., 2021] Mata, J. M. S., Navarro, C. C., Sánchez, A. V. B., and Uribe, H. F. D. (2021). La seguridad informática en las instituciones de educación superior. *Revista de divulgación científica y tecnológica*, 7(2):72. Recibido: 24/09/2021, Aceptado: 6 de noviembre de 2021.

[Mitre,] Mitre. Common weakness enumeration.

[Myers et al., 2011] Myers, G., Sandler, C., and Badgett, T. (2011). *The Art of Software Testing*. ITPro collection. Wiley.

[OWASP, 2021] OWASP (2021).

[Pressman, 2010] Pressman, R. (2010). *INGENIERIA DE SOFTWARE*. McGraw-Hill Interamericana de España S.L.

[Pérez-Gavilán et al., 2018] Pérez-Gavilán, J. J., Aguirre, J., and Ramírez, L. (2018). Sismicidad y seguridad estructural en las construcciones: lecciones aprendidas en méxico. *Salud Pública de México*, 60(Supl. 1):41–51.

[Socas Gutiérrez and Déniz, 2023] Socas Gutiérrez, R. and Déniz, L. G. (2023). *Redes de ordenadores: Principios y aplicaciones para la ingeniería del software*. Marcombo, España.

[Sommerville, 2005] Sommerville, I. (2005). *Ingeniería del software*. Pearson Educación, Spain.

[Stallings, 2004] Stallings, W. (2004). *Fundamentos de seguridad en redes: aplicaciones y estándares*. Pearson, Prentice Hall, España.

[Sánchez, 2024] Sánchez, A. G. (2024).

[Tanenbaum, 2003] Tanenbaum, A. S. (2003). *Redes de computadoras*. Editorial Alhambra S. A. (SP), España.

[Web Accessibility Initiative (WAI), 2025] Web Accessibility Initiative (WAI) (2025). WCAG 2.1 de un vistazo. `https://www.w3.org/WAI/standards-guidelines/wcag/glance/es`. Consultado el [fecha de acceso].