ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERIA DE SISTEMAS

UNIDAD DE TITULACIÓN

Aplicación y análisis de rendimiento de bases de datos cifradas para el almacenamiento seguro de registros médicos electrónicos: Caso de estudio CryptDB.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SOFTWARE MENCIÓN SEGURIDAD

Ing. Fabián Santiago García Bautista

santiagogarciabautista@gmail.com

Director: PhD. Gabriela Lorena Suntaxi Oña

gabriela.suntaxi@epn.edu.ec

2023

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación “Aplicación y análisis de rendimiento de bases de datos cifradas para el almacenamiento seguro de registros médicos electrónicos: Caso de estudio CryptDB.” desarrollado por Fabián Santiago García Bautista, estudiante de la Maestría en Software Mención Seguridad, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

PhD. Gabriela Lorena Suntaxi Oña

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Fabián Santiago García Bautista, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Firmado electrónicamente por:
FABIAN SANTIAGO
GARCIA BAUTISTA

Fabián Santiago García Bautista

DEDICATORIA

A mi amada esposa, quien ha estado a mi lado en cada paso de este viaje. Su apoyo inquebrantable y su amor incondicional me han dado la fuerza para superar innumerables desafíos. Este trabajo no hubiera sido posible sin su paciencia y comprensión.

A mi madre, cuya insistencia y convicción en mi potencial nunca flaquearon. Gracias por alentarme a perseguir esta maestría y por creer en mí incluso cuando dudaba de mí mismo. Esta realización es tan mía como suya.

A mis leales compañeros, Loky y Kony, quienes me han proporcionado una fuente constante de consuelo y alegría. Su presencia ha sido un recordatorio diario de la simplicidad y la belleza de la vida.

En memoria de mi amado Dachi, a quien dedico este trabajo. Aunque ya no está físicamente con nosotros, su espíritu vive en cada página de este trabajo. Dachi, tu recuerdo sigue siendo una fuente constante de inspiración.

Y no podría olvidarme de mi tutora de tesis,Gabriela, cuya paciencia y guía han sido invaluables en este viaje. Su disposición a esperar y su inquebrantable fe en mi capacidad para completar este trabajo han sido cruciales para llegar a este punto. Gracias por su sabiduría, su apoyo y su persistencia constante en mi éxito. Su influencia ha dejado una huella indeleble en mi carrera académica, profesional y en mi vida.

Y finalmente, dedico este trabajo a mí mismo. En los momentos de duda, decidí seguir adelante. En los momentos de incertidumbre, tomé las decisiones correctas. Este trabajo es un testimonio de la determinación y la resistencia que descubrí en mí. Es un recordatorio de que, sin importar las adversidades que enfrentemos, siempre podemos superarlas y alcanzar nuestras metas.

Índice general

Índice de figuras

Indice de tablas

RESUMEN

En este artículo, exploramos la implementación de CryptDB en un ambiente productivo, enfocándonos especialmente en el rendimiento. A través de la aplicación del escalamiento horizontal, hemos logrado reducir la latencia hasta en un 50 % en las operaciones de lectura, lo cual representa un avance significativo hacia la posibilidad de utilizar CryptDB en entornos reales. No obstante, este beneficio no se extiende a las operaciones de escritura, ya que la complejidad del cifrado de datos presenta desafíos adicionales. Para abordar esta limitación, recomendamos la implementación de la arquitectura Command Query Responsibility Segregation (CQRS), que permite separar y optimizar las operaciones de lectura y escritura. Aunque estos resultados son prometedores, se deben considerar otros factores como la seguridad y la compatibilidad en la evaluación integral de la incorporación de CryptDB. El artículo destaca tanto los avances logrados como los desafíos que aún deben abordarse para una integración exitosa.

ABSTRACT

This study presents an investigation into the performance of CryptDB, a database system providing practical and provable confidentiality in the face of an adversary who captures all stored data, under different cloud-based architectures. The research aims to determine how different configurations impact the scalability and efficiency of CryptDB. The cloud architectures examined include an encrypted environment with no scaling (EENS), an encrypted environment with scaling (EEWS), and an unencrypted data (UE) architecture. A load testing tool, k6, is used to simulate various concurrent user scenarios. The results demonstrate that the proposed EEWS architecture can reduce latency by up to 50 %, though it also highlights CryptDB's limitations, including poor CPU utilization and inadequate documentation. Recommendations are provided for future enterprise projects considering CryptDB. Further research can expand upon this work by considering other alternative tools and their performance under high concurrency.

Capítulo 1

INTRODUCCIÓN

1.1. Motivación

Uno de los problemas que enfrentan las empresas al tener una base de datos en las instalaciones (On-Premises) o en la nube (On-Cloud) es la privacidad de sus datos. Los datos de las empresas están expuestos a amenazas externas e internas. Las amenazas externas, por lo general, están fuera de la red y son perpetradas por agentes cuyo objetivo es apoderarse, alterar o eliminar la información mediante la identificación y explotación de vulnerabilidades. Las amenazas internas provienen de actores que forman parte de la organización, es decir son agentes internos que tienen acceso privilegiado a la información, como por ejemplo administradores de bases de datos o de sistemas[1]. Un atacante interno que tiene acceso a los datos, puede leer los mismos y usarlos de forma mal intencionada [2]. A estos tipos de atacantes se los conoce en la literatura como curiosos o maliciosos.

Cuando se deposita información en un sistema que está alojado principalmente en la nube, se debe ofrecer garantías de que la información está segura, especialmente si los datos almacenados son sensibles como por ejemplo datos médicos. En este caso, los cibercriminales podrían beneficiarse financieramente del robo de dicha información. Por ejemplo, la información robada podría ser vendida a terceros, quienes podrían realizar un análisis para identificar individuos de alto riesgo en el ámbito de un seguro de salud debido a su historial médico o desórdenes genéticos. Por lo tanto, el almacenamiento de registros médicos o de salud también conocidos en inglés como “Electronic Health Record(EHR)” o “Electronic Medical Record(EMR)”, deben cumplir regulaciones de privacidad como el cifrado de los datos y control de acceso estricto. Estos controles de seguridad deben garantizar el cumplimiento de los tres objetivos fundamentales de la seguridad que son [3]: confidencialidad, integridad y disponibilidad. De

acuerdo a la norma ISO 13606[4], la confidencialidad se refiere al proceso que asegura que la información es accesible solo a quienes fueron autorizados a tenerla. La integridad significa asegurar que la información es precisa y no ha sido alterada de manera no autorizada. La disponibilidad significa que la información debe ser accesible y utilizable a demanda por un ente autorizado[4].

Para cumplir con estándares de seguridad de datos como lo son "Payment Card Industry Data Security Standard (PCI DSS)", "Health Insurance Portability and Accountability Act (HIPAA)ƨ "General Data Protection Regulation (GDPR)", algunos fabricantes de sistemas de manejo de base de datos (DBMS) ofrecen un servicio de cifrado de datos llamado "Tranparent Data Encryption (TDE)", el cual es ofrecido por reconocidos DBMS como MySQL [5], Oracle Database [6] y SqlSever[7]. TDE es usado para la encriptación y desencriptación por medio del uso de una llave de encriptación de base de datos (DEK)[8]. Esta llave es asimétrica y está protegida por un certificado almacenado en una base de datos maestra [8]. Este tipo de encriptación agrega una capa de seguridad garantizando la confidencialidad de los datos[8] [2]. Sin embargo, el servicio TDE solo se encuentra disponible en versiones comerciales de paga como lo son en MySQL Enterprise, SqlServer y Oracle Database. Lo cual limita su utilización, especialmente en proyectos de bajo presupuesto.

Por lo tanto, es necesario investigar y analizar la factibilidad de utilizar herramientas open source que permitan garantizar la confidencialidad de los datos sin incurrir en gastos adicionales para las organizaciones que han decidido migrar sus datos a la nube. El uso de estas herramientas ayudará a las organizaciones a cumplir con estándares de seguridad y privacidad de datos.

1.2. Definición del Problema y Dificultades

En la literatura existe una variedad de herramientas que permiten cifrar los datos de las bases de datos, ya sea que se encuentren en tránsito o en reposo. Entre ellas tenemos CryptDB. Los autores en [2] han propuesto un sistema llamado CryptDB, el cual es un proxy de conexión entre la aplicación y la base de datos que permite tener un cifrado de los datos en tránsito.

Los autores en [9] encontraron que los tiempos de respuesta se incrementan considerablemente cuando se utiliza CryptDB. El presente trabajo busca verificar la factibilidad de implementar CryptDB en entornos productivos que requieran que su tiempo de respuesta sea bajo. Para lo cual, se comparará diferentes arquitecturas y se propondrá mejoras que permitan disminuir los tiempos de respuesta. La solución a implementarse debe mantener la seguridad de los datos usando CryptDB, pero sin incrementar los costos de manera significativa.

Existen diferentes arquitecturas que podrían implementarse y es necesario realizar una investigación experimental que permita evaluar que modelo arquitectónico produce mejores resultados en cuanto a rendimiento. La arquitectura a sugerirse podría apoyarse en un escalamiento horizontal, ya que este tipo de escalamiento permite ahorrar costos y hacerlo automáticamente cuando cumpla cierto parámetros. Sin embargo, es necesario un estudio experimental que permita evaluar las mejoras en cuanto a rendimiento.

Otro limitante que puede afectar el proceso de implementación es que CryptDB, según la documentación oficial, ha sido probado en el sistema operativo Ubuntu v12.04 y v13.04, los cuales son relativamente antiguos. Por lo tanto, es necesario verificar si CryptDB puede ser implementado en sistemas actuales o de ser necesario encontrar un sistema adecuado en la nube.

Además de lo expuesto anteriormente, existe la posibilidad que CryptDB no sea compatible con las versiones disponibles de MySQL, lo cuál obligaría a crear un servidor e instalar MySQL de versiones anteriores para poder realizar los experimentos deseados.

Es importante realizar un análisis experimental para verificar que la solución propuesta es la adecuada y cabe dentro de los parámetros de aceptación. Este análisis podría incurrir en gastos que podrían llegar a ser un limitante, si se dispone un presupuesto apretado. El análisis experimental se llevará acabo en Amazon Web Services (AWS), el cual dispone de varios servicios. Conocer todos los servicios que soporta este proveedor no es una tarea trivial. Por lo tanto, se requiere experiencia en el manejo de estos servicios que permita aportar conocimiento a la implementación.

Otro aspecto a considerar, es la selección de las métricas de evaluación, las cuales se seleccionan en primera instancia en base a supuestos que podrían cambiar a lo largo del trabajo y podría ocasionar tareas repetitivas las cuales podrían afectar el presupuesto. En [9] se utiliza solo las métricas de CPU de utilización de memoria RAM, lo cuál deja otros recursos sin analizar como uso de disco de almacenamiento o red. Este trabajo busca cubrir otras métricas que afectan el rendimiento como Operaciones de Lectura por Segundo, Operaciones Promedio Encoladas, Operaciones de Escritura por Segundo, entre otras.

1.3. Objetivo General

Analizar el rendimiento del sistema CryptDB para el almacenamiento seguro de registros médicos electrónicos de manera cifrada en la Nube.

1.4. Objetivos Específicos

- Analizar mediante estudios experimentales y con una base de datos que contiene datos reales la arquitectura que utiliza CryptDB para determinar problemas de rendimiento.
- Proponer una posible solución en nivel de arquitectura para los cuellos de botella que se pueden encontrar cuando se utiliza CryptDB en consulta, escritura y actualización de datos.
- Entender si CryptDB puede ser integrado efectivamente en un entorno empresarial, identificar posibles desafíos y limitaciones, y sugerir estrategias para mitigar estos.

1.5. Contribución

Esta trabajo tiene como propósito, el proponer una arquitectura en la nube en la cual CryptDB se pueda utilizar en ambientes productivos, para proyectos de bajo

presupuesto los cuales no estén en capacidad de obtener un software propietario. Se contribuye con un diseño de arquitectura en la cual se verifica que CryptDB se puede utilizar hasta cierto límite. El análisis ayuda a comprender las limitaciones de CryptDB en su estado actual.

A continuación se presenta de manera sintetizada la estructura del presente trabajo de titulación.

En el Capítulo 1 “Introducción” se presenta de manera breve la motivación del presente trabajo. Se describe la razón de trabajar en un tema como “Aplicación y análisis de rendimiento de bases de datos cifradas para el almacenamiento seguro de registros médicos electrónicos: Caso de estudio CryptDB.”. Además, se relata cual es es problema que se va a cubrir y enumera los objetivos del trabajo. Adicionalmente, se menciona la contribución que se realiza a la comunidad científica y de profesionales.

En el Capítulo 2 “Marco Teórico” se incluye los fundamentos teóricos que envuelven y aportan al trabajo actual, como cifrado o encriptación, encriptación homomórfica y su aporte en el actual trabajo. Se revisa la teoría detrás de CryptDB y sus tipos de encriptación. También se incluyen algunos servicios de Amazon Web Services (AWS) que van a ser usados y se explica de forma global su funcionamiento.

En el Capítulo 3 “Metodología” se explica el método y la metodología detrás del proyecto de investigación. Se expone cuales son las métricas que intervienen en los factores de decisión para mejoramiento de la arquitectura. Además, se expone los pasos a seguir y poner en marcha durante la investigación.

En el Capítulo 4 “Resultados y Discusión” se expone los resultados en base a la metodología aplicada y se justifica la selección de las arquitecturas adoptadas que permiten alcanzar un tiempo de respuesta bajo.

En el Capítulo 5 “Conclusiones y Recomendaciones” se analiza el potencial de las arquitecturas adoptadas y sugeridas, y se presenta las conclusiones finales del trabajo desarrollado. Del mismo modo se realizan recomendaciones en base a los resultados obtenidos.

1.6. Revisión sistemática de la literatura sobre CryptDB

La metodología de Kitchenham se ha seleccionado para llevar a cabo la Revisión Sistemática de la Literatura (RSL) en este estudio debido a su alto nivel de validez en la realización de investigaciones. Este enfoque nos permitirá obtener una visión completa y replicable sobre CryptDB, su rendimiento y su aplicabilidad en diferentes arquitecturas de nube. En la sección 3.2 se encuentra una amplia explicación de la metodología.

1.6.1. Planificar la Revisión

Esta fase considera tres puntos clave en la realización de una RSL: la especificación de las preguntas de investigación, la definición de los criterios de inclusión y exclusión, y la selección de bases de datos y cadenas de búsqueda.

1.6.1.1. Especificación de las Preguntas de Investigación

Las preguntas de investigación formuladas para guiar esta RSL son:

- QI1: ¿Qué arquitecturas de nube son compatibles con CryptDB?
- QI2: ¿Cómo afecta la configuración de CryptDB al rendimiento y la escalabilidad en diferentes arquitecturas de nube?
- QI3: ¿Qué limitaciones han sido identificadas en la implementación y uso de CryptDB?
- QI4: ¿Con qué tipo de arquitectura/ proveedores de la nube se ha implementado CryptDB?

1.6.1.2. Definir Criterios de Inclusión y Exclusión

Criterios de Inclusión:

- Artículos que aborden el rendimiento y la escalabilidad de CryptDB, esto debe ser mencionado en su título o en el abstract.

- Estudios que examinen CryptDB en diferentes arquitecturas de nube.
- Trabajos publicados en revistas, conferencias o editoriales de alto impacto.
- Artículos en inglés, dado que es el idioma dominante en esta área de investigación.
- Artículos citados al menos una vez.
- Artículos que tengan en el título “CryptDB”

Criterios de Exclusión:

- Artículos no relacionados directamente con CryptDB o arquitecturas de nube.
- Documentación y manuales de usuario de CryptDB.
- Trabajos con más de 7 años de antigüedad.

1.6.1.3. Definición de las Bases de Datos y Cadenas de Búsqueda

Se utilizarán las siguientes bases de datos para la revisión:

- IEEE Xplore
- ACM Digital Library
- Google Scholar
- DBLP
- ScienceDirect

Cadenas de Búsqueda:

1. “CryptDB” AND “Cloud” AND “Performance”
2. “CryptDB” AND “Scalability” AND “Configuration”

1.6.2. Conducir la Revisión

En esta fase, se llevarán a cabo cinco subfases: Identificación de fuentes/estudios relevantes, selección de estudios primarios, evaluación de la calidad de los estudios, extracción de los datos requeridos y síntesis de los datos.

1.6.2.1. Identificar Fuentes/Estudios Relevantes

Siguiendo los criterios de inclusión y exclusión, y utilizando las cadenas de búsqueda en las bases de datos seleccionadas, se identificarán los estudios más relevantes. Google Scholar se seleccionó como el motor de búsqueda principal para esta etapa de la investigación. La razón detrás de esta elección radica en que Google Scholar tiene la capacidad de indexar y buscar a través de los repositorios que se mencionan en la sección 1.6.1.3 del documento.

Resultados de la Primera Cadena de Búsqueda: En la Figura 1.1, se muestran los hallazgos de la Primera Cadena de Búsqueda. El último término de búsqueda aplicado en Google Scholar fue:

```
"Performance" "Cloud" intitle:CryptDB
```

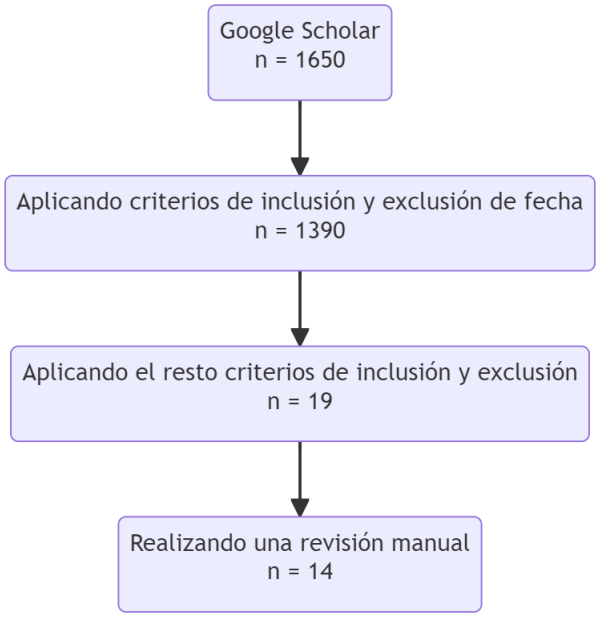


Figura 1.1: Resultados de la Primera Cadena de Búsqueda

Resultados de la Segunda Cadena de Búsqueda: En la Figura 1.2, se muestran los hallazgos de la Segunda Cadena de Búsqueda. El último término de búsqueda aplicado en Google Scholar fue:

```
"Scalability" "Configuration" intitle:CryptDB
```

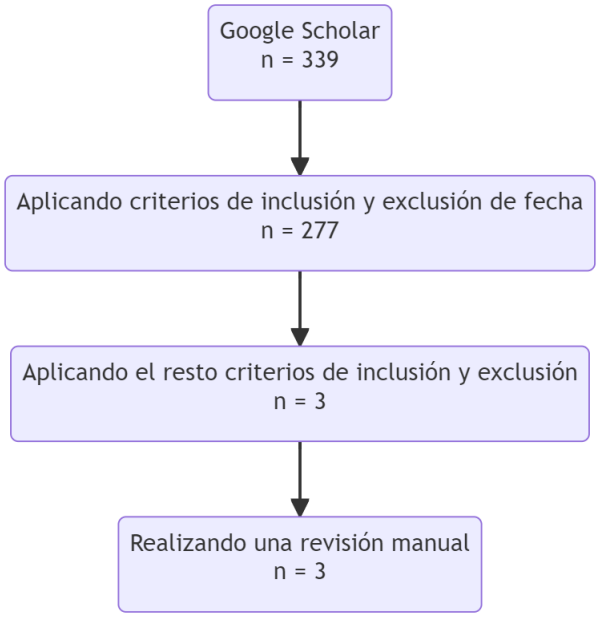


Figura 1.2: Resultados de la Segunda Cadena de Búsqueda

En esta última, el número de resultados es mucho menor lo cuál ayuda a validar que la configuración de CryptDB no juega un rol muy importante en la investigación.

1.6.2.2. Seleccionar Estudios Primarios

Los estudios que pasen los criterios se catalogarán como estudios primarios.
El cuadro 1.1 y el cuadro 1.2 es un resumen detallado y organizado de los artículos seleccionados a través de la metodología de Kitchenham, y sirve para sintetizar la información relevante de cada uno de ellos. La tabla está compuesta por las siguientes seis columnas:

- **ID**: Esta columna contiene un identificador único para cada artículo incluido en el estudio, con el fin de facilitar su posterior referencia.

- **Autor**: Aquí se listan los nombres de los autores del artículo, proporcionando contexto sobre quiénes son los expertos detrás del trabajo de investigación.

- **Año de Publicación**: Indica el año en el cual el artículo fue publicado. Aunque la fuente principal de información es Google Scholar, la columna se titula simplemente “Año” para una mejor organización visual de la tabla.

- **Tópico** : Esta columna describe el tema a tratar.

- **Tipo de Artículo**: En esta columna, se especifica el tipo del documento, es decir, si es un artículo de revista, un libro, una tesis, entre otros.

- **Calidad del Artículo**: La calidad del artículo se evalúa según los criterios establecidos en el estudio “Systematic Review of Issues and Solutions for Security in E-commerce” 14.

Tabla 1.1: Resultados de la Primera Cadena de Búsqueda

ID	Autor	Año	Tópico	Tipo de Artículo
1	Shahzad et al.	2015	Estudio de rendimiento	Conferencia
2	Yousuf et al.	2021	Revisión sistemática	Journal
3	Zhuang et al.	2017	Estudio de rendimiento	Conferencia
4	Yao & Xue	2018	Aceleración de algoritmos	Conferencia
5	Jiang et al.	2020	Estudio de esquemas	Conferencia
6	Foltz & Simpson	2018	Extensión de CryptDB	Conferencia
7	Popa et al.	2015	Directrices de uso	Journal
8	Nasereddin & Darwesh	2020	Programación orientada a objetos	Journal
9	Rauthan & Vaisla	2017	Análisis computacional	Conferencia
10	Skiba et al.	2015	Análisis de bases de datos	Tesis de pregrado
11	Kumar & Hussain	2018	Procesamiento de consultas	Conferencia
12	Aburawi	2020	Mecanismo en bases de datos de grafos	Tesis doctoral
13	Maisura	2018	Análisis de cifrado del lado del cliente	Journal
14	Lopez & Chua	2017	Seguridad en Sistemas de Salud	Conferencia

Tabla 1.2: Resultados de la Segunda Cadena de Búsqueda

ID	Autor	Año	Tópico	Tipo de Artículo
1	Jiang et al.	2020	Estudio de esquemas	Conferencia
2	Shahzad et al.	2015	Estudio de rendimiento	Conferencia
3	Foltz & Simpson	2018	Extensión de CryptDB	Conferencia

1.7. Extracción de Datos Requeridos

El objetivo de esta sección es resumir los hallazgos clave de la literatura existente que son relevantes para mejorar el rendimiento de CryptDB, particularmente cuando se implementa en entornos de nube.

1.7.1. Rendimiento en la Nube y en la Salud

Shahzad et al. [10] se enfocaron en el rendimiento de CryptDB para asegurar los datos de salud electrónicos en la nube. Su estudio reveló que CryptDB podría asegurar efectivamente los datos de salud con una sobrecarga mínima en sistemas basados en la nube.

1.7.2. Revisión Sistemática de CryptDB

Yousuf et al. [11] proporcionaron una revisión exhaustiva sobre CryptDB, discutiendo sus desafíos de implementación y oportunidades futuras. La revisión indicó que, aunque CryptDB tiene sus ventajas, todavía hay oportunidades para mejorar su rendimiento, especialmente en arquitecturas de nube.

1.7.3. Aceleración AES-NI

Zhuang et al. [12] sugirieron el uso de AES-NI (Advanced Encryption Standard New Instructions) para mejorar el rendimiento de CryptDB. Esta solución basada en hardware podría ser especialmente efectiva en entornos de nube que soportan AES-NI.

1.7.4. Teorema del Resto Chino

Yao y Xue [13] exploraron acelerar el criptosistema Paillier en CryptDB usando el Teorema del Resto Chino. Esta mejora algorítmica podría aumentar significativamente el rendimiento, lo que lo hace altamente aplicable para soluciones de nube escalables.

1.7.5. Cifrado que Preserva el Orden

Jiang et al. [14] presentaron una investigación sobre esquemas de cifrado que preservan el orden basados en CryptDB. Este esquema podría mejorar el rendimiento de las consultas, lo cual es particularmente útil en escenarios de nube donde la latencia es una preocupación.

1.7.6. Extendiendo CryptDB

Foltz y Simpson [15] examinaron la extensión de CryptDB para operar sistemas de planificación de recursos empresariales (ERP) en datos cifrados. Su trabajo sugiere que CryptDB se puede escalar y adaptar para diferentes dominios de aplicación, incluidos los sistemas basados en la nube.

1.7.7. Pautas de Seguridad

Popa et al. [16] proporcionaron pautas para usar el sistema CryptDB de forma segura. Aunque no está directamente relacionado con el rendimiento, una configuración segura es vital para un rendimiento optimizado, especialmente en implementaciones en la nube.

1.7.8. Otros Hallazgos

Varios otros artículos [17]-[23] exploraron diversos aspectos de CryptDB, desde la programación orientada a objetos hasta el procesamiento de consultas, que en conjunto contribuyen a entender cómo se puede mejorar el rendimiento de CryptDB en un entorno de nube.

Síntesis de los Datos

Los estudios revisados aportan una visión comprensiva pero fragmentada de las capacidades, el rendimiento y las limitaciones de CryptDB en distintas arquitecturas de nube. Haciendo énfasis a las preguntas de investigación formuladas

Arquitecturas de Nube Compatibles (QI1): Aunque los detalles específicos son escasos, es evidente que CryptDB puede implementarse en diversas arquitecturas de nube. Shahzad et al. [10] ilustran su aplicabilidad en un entorno de nube centrado en el almacenamiento seguro de registros de salud electrónicos.

Impacto en el Rendimiento y Escalabilidad (QI2): Zhuang et al. [12] y Yao et al. [13] sugieren que el rendimiento de CryptDB puede mejorarse significativamente mediante la implementación de técnicas de aceleración, como AES-NI y el Teorema del Resto Chino. Foltz et al. [15] demostraron que incluso sistemas empresariales complejos como los ERP pueden operar de manera escalable utilizando CryptDB, aunque el estudio no aborda métricas de rendimiento específicas.

Limitaciones Identificadas (QI3): Aunque CryptDB tiene ventajas significativas en términos de seguridad, Yousuf et al. [11] identifican que hay un compromiso entre seguridad y rendimiento que debe ser manejado cuidadosamente. La literatura es menos exhaustiva en la discusión de las limitaciones relacionadas con la escalabilidad y otras métricas de rendimiento.

Implementación con Proveedores de Nube (QI4): Sorprendentemente, hay una falta de información específica sobre qué proveedores de nube han implementado CryptDB en sus plataformas de muchos artículos. Solo [10] habla que implementó las pruebas en AWS como proveedor Cloud. Esto representa una oportunidad para futuras investigaciones, ya que entender las compatibilidades y desafíos específicos con diferentes proveedores podría ofrecer orientaciones más precisas para la adopción de CryptDB en la industria.

En resumen, aunque CryptDB ofrece soluciones robustas para la seguridad de bases de datos [16], el rendimiento y la escalabilidad son aspectos que pueden ser mejorados mediante configuraciones y optimizaciones específicas [18]. Se necesita más investigación para ofrecer una visión completa de cómo estos factores interactúan en distintas arquitecturas de nube y qué proveedores de servicios de nube ofrecen el entorno más idóneo para maximizar las capacidades de CryptDB [21].

1.8. Conclusión de la Revisión Sistemática de la Literatura

La revisión sistemática de la literatura ha proporcionado una visión global sobre el estado del arte en el uso de CryptDB para bases de datos cifradas, con un enfoque particular en el almacenamiento seguro de registros médicos electrónicos. Aunque CryptDB emerge como una solución potente en términos de seguridad [16], varios estudios han señalado que existen áreas para mejorar su rendimiento [12], [13].
En el contexto específico de los registros médicos electrónicos, Shahzad et al. demostraron que CryptDB es una herramienta viable [10]. Sin embargo, es crucial entender que existen compromisos entre la seguridad y el rendimiento que requieren una gestión cuidadosa [11].
La literatura actual presenta una falta notable de estudios que analicen la compatibilidad y el rendimiento de CryptDB en diferentes proveedores de servicios de nube [21]. Solo Shahzad et al, probó y realizó las pruebas en AWS. Esta es una oportunidad abierta para futuras investigaciones que podrían ofrecer orientaciones valiosas para la adopción de CryptDB en contextos sanitarios y en la Nube.
Resumidamente, el rendimiento y la escalabilidad son aspectos que pueden y deben ser mejorados para maximizar la eficacia de CryptDB. La optimización del rendimiento no solo contribuirá a una adopción más extensa de CryptDB, sino que también permitirá una implementación más eficiente y eficaz en entornos que manejan datos sensibles como registros médicos electrónicos [18].
Nuestra tesis buscará abordar algunas de estas brechas, enfocándose en la evaluación del rendimiento de CryptDB en diferentes arquitecturas de nube y su idoneidad para el almacenamiento seguro de registros médicos electrónicos.

Capítulo 2

MARCO TEÓRICO

En este capítulo se exponen cuatro secciones. La primera sección presenta el cifrado y los tipos de cifrado que se utilizarán en el presente trabajo. En la segunda sección se presenta CryptDB y sus características en conjunto con su funcionalidad y se explica como CryptDB logra prevenir ciertos tipos de riesgos. La tercera sección presenta la tecnología de nube que se utilizará y que se la conoce como Amazon Web Services(AWS). Además, se menciona los servicios que se usarán en el presente trabajo. Por último en la cuarta sección se trata sobre la base de datos que se utilizará en la fase de evaluación del presente proyecto y se explica como se obtuvieron los datos.

2.1. Cifrado

El cifrado o encriptación es el proceso de convertir mensajes o datos en un mensaje no legible por nadie más, excepto por la persona para quien que fue destinado el mensaje. Los datos encriptados deben ser desencriptados antes de ser leídos. En anteriores tiempos, la encriptación se lo hacia por medio de substitución, es decir se substituían partes de la información con símbolos, números o imágenes.

El objetivo principal del cifrado es mantener la confidencialidad e integridad de la información. En las comunicaciones de la actualidad, debido a la sensibilidad de los datos es imprescindible el implementar un cierto nivel de cifrado.

Hoy en día se utilizan algoritmos avanzados de encriptación los cuales están divididos en dos categorías:

- *Algoritmos simétricos:* Este tipo de algoritmos conocidos también como algoritmos de llave privada, usan la misma llave para encriptar y desencriptar el contenido. Actualmente uno de los más usados es el AES(Advanced Encryption Standar) conocido también como Rijndael.

- *Algoritmos asimétricos:* Este tipo de algoritmos también conocidos como algoritmos de llave pública, usan la llave pública para encriptar el mensaje y la llave privada para desencriptar el mensaje. Actualmente uno de los más usados es el RSA (Rivest–Shamir–Adleman).

Estos algoritmos de estás dos categorías pueden ofrecer diferentes niveles de seguridad dependiendo del tipo de cifrado que utilicen: determinista o probabilista.

2.1.1. Cifrado Determinista

Este tipo de cifrado se caracteriza por producir el texto cifrado repetidamente cuando se utilice la misma llave y texto plano. Por ejemplo, con el texto plano *"La criptografía es grandiosa"* el texto cifrado resultante siempre va a ser *"Zo qfwdhcufotío sg ufobrwcgo"* siempre y cuando se mantenga la misma llave y el mismo texto plano. Para el ejemplo se utilizó uno de los algoritmos más antiguos conocido como “Caesar cipher” con una llave de desplazamiento de 14 caracteres. En la Figura 2.1 se puede observar gráficamente como funciona el proceso de cifrado donde M1 es el mensaje y C1 es el texto cifrado resultante.

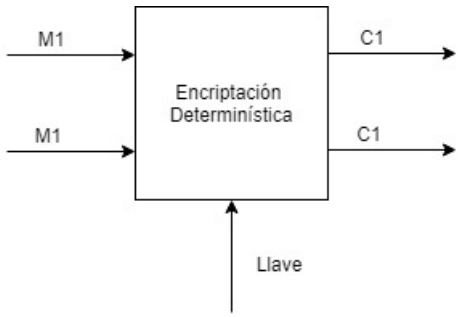


Figura 2.1: Encriptación Determinista

2.1.2. Cifrado Probabilístico

Este tipo de cifrado se caracteriza por producir el texto cifrado diferente cada vez que se realiza el cifrado dado la misma llave y mismo texto plano. Por ejemplo, con el texto plano *"La criptografía es grandiosa"* el texto cifrado resultante será siempre diferente cada vez que sea realice el cifrado. Este tipo de cifrado brinda

un nivel superior de cifrado y seguridad semántica porque introduce un contenido aleatorio al contenido original y gracias a esto logra producir diferentes valores en el proceso de cifrado. En la Figura 2.2 se puede observar gráficamente como funciona el proceso donde M1 es el mensaje y C1 y C2 son los textos cifrados resultantes.

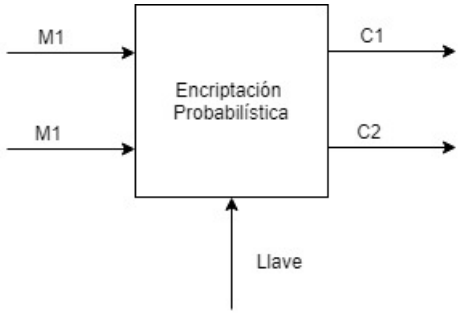


Figura 2.2: Encriptación Probalística

2.1.3. Cifrado Homomórfico

El cifrado o encriptación homomórfica puede ser visto como una extensión del cifrado asimétrico. Este cifrado se lo considera probabilístico para su encriptación utilizando algoritmos asimétricos. Existen varios tipos de algoritmos asimétricos los cuales producen una cifrado homomórfico como es el caso de RSA(Rivest–Shamir–Adleman) clásico o del esquema The Goldwasser-Micali.

Este cifrado es un tipo de encriptación que permite llevar acabo procesamiento computacional sobre texto cifrado y cuyo resultado va a ser el mismo si el procesamiento se lo realizaba sobre texto plano[24]. Este tipo de cifrado es probabilístico.

RSA(Rivest–Shamir–Adleman) es un algoritmo de cifrado de llave pública que tiene una propiedad homomórfica. Al momento de añadir un "padding.º también conocido como relleno, pierde su propiedad homomórfica; esto se realiza con el objetivo de lograr seguridad semántica en la encriptación [24].

En álgebra, el homomorfismo representa la conservación de la estructura de un mapa entre dos estructuras. Es decir, como se puede ver en la Figura 2.3 los dos

grupos tienen la misma forma y se puede realizar operaciones con esos grupos y los resultados serán equivalente entre los grupos.

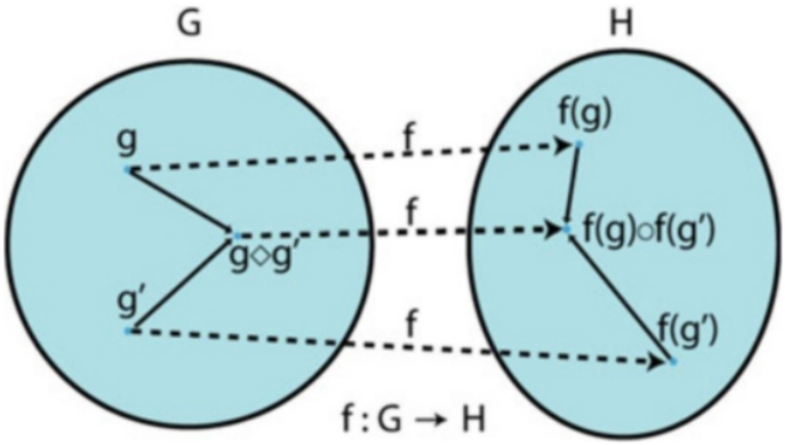


Figura 2.3: Homomorfismo [25]

2.2. CryptDB

CryptDB es una solución OpenSource diseñada para ofrecer una capa adicional de seguridad en sistemas de gestión de bases de datos (DBMS) como MySQL Community Edition o Postgres. Actúa como un proxy de conexión entre la aplicación y la base de datos, permitiendo que los datos se cifren tanto en reposo como en tránsito [2].

El propósito principal de CryptDB es garantizar la confidencialidad de la información. Protege los datos de atacantes internos o externos que puedan obtener acceso a la base de datos, evitando que puedan aprender o extraer datos privados. Para lograr esto, CryptDB implementa tres ideas clave [2]:

- **Ejecución de Consultas Cifradas:** Permite realizar consultas en datos cifrados, sin necesidad de descifrarlos, lo que añade una capa adicional de seguridad.
- **Cifrado Dinámico Ajustable:** Mediante el análisis de las consultas en tiempo de ejecución, CryptDB puede decidir qué algoritmo de cifrado

implementar sobre las columnas, permitiendo un equilibrio entre seguridad y eficiencia.

- **Vinculación de Claves de Cifrado:** Asocia las claves de cifrado con las contraseñas de los usuarios, asegurando que sólo los propietarios legítimos de la información puedan descifrar el texto cifrado.

Además de la confidencialidad del contenido, CryptDB también protege los nombres de las columnas y las tablas de la base de datos.
Sin embargo, CryptDB tiene algunas limitaciones. No esconde la estructura, el número y el tipo de las columnas, y revela al servidor de base de datos las relaciones entre los datos al ejecutar cálculos computacionales para consultas como comparación, equidad, ordenamiento y búsqueda de palabras [2].
La seguridad de CryptDB no es perfecta, pero ofrece una mejora significativa en la protección de datos sensibles. La información revelada solo dependerá del tipo de relaciones en las consultas realizadas por la aplicación que necesite la información. A pesar de sus limitaciones, CryptDB es una herramienta valiosa que garantiza que los datos sensibles nunca estarán disponibles en texto plano para el servidor de base de datos(DBMS), fortaleciendo así la postura de seguridad de una organización [2].

2.2.1. Funcionamiento de las consultas SQL en CryptDB

Las consultas se basan en un esquema donde la aplicación y el proxy son confiables. Es decir, el ambiente es controlado y determinado como seguro por el dueño de la aplicación y por el contrario, el DBMS y su administrador no son confiables [2].
CryptDB le permite al servidor de la base de datos ejecutar consultas SQL como si lo estuviera haciendo en datos planos y funciona mediante el almacenamiento de una llave maestra llamada "MK"(Master Key). Además, permite definir funciones definidas por usuario (UDF's). El procesamiento de las consultas se lo realiza a través de cinco pasos:

1. La aplicación realiza una consulta SQL, la cuál es interceptada por el proxy y la reescribe.

2. El proxy anonimiza cada tabla y nombre de columna utilizando la llave maestra (MK), encripta cada constante en la consulta con el esquema de encriptación que mejor se ajuste para la operación solicitada.

3. El proxy verifica si el DBMS necesita obtener llaves para ajustar las capas de encriptación antes de ejecutar la consulta SQL. Si es así, el proxy ejecuta una consulta UPDATE al DBMS que invoca a Funciones Definidas por el Usuario(UDF) para ajustar la capa de encriptación para las columnas apropiadas.

4. El proxy envía la consulta SQL al servidor DBMS que ejecuta una consulta SQL estándar.

5. El servidor DBMS retorna el resultado que se encuentra encriptado, el proxy lo desencripta y lo retorna a la aplicación.

Los pasos descritos en el párrafo anterior, se pueden visualizar de manera gráfica en la Figura 2.4

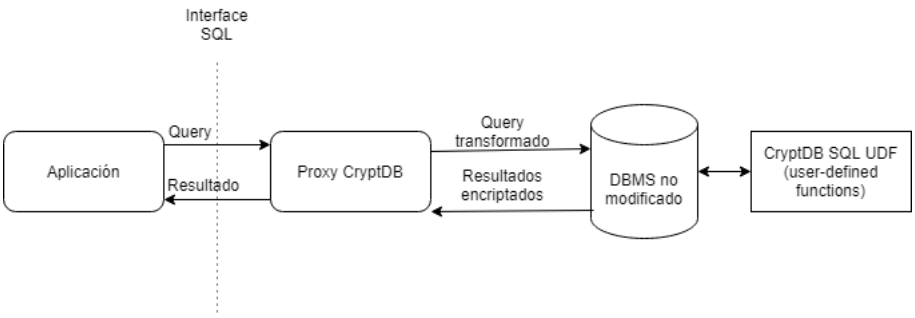


Figura 2.4: Flujo de datos en CryptDB

2.2.2. Tipos de Encriptación de CryptDB

CryptDB implementa diferentes tipos de encriptación según el tipo de operación requerida en la consulta SQL.

Random(RND) Este esquema si bien no permite realizar cálculos de forma eficiente sobre el texto cifrado, brinda el más alto nivel de seguridad ya que tiene un esquema probabilístico. Es decir, que dos textos planos iguales son mapeados a dos textos cifrados diferentes.

Deterministic (DET) Este esquema permite realizar cálculos de equidad como GROUP BY, COUNT, DISTINCT, etc., ya que dos textos planos iguales van a dar dos textos cifrados iguales. Este algoritmo es menos seguro con relación a RND.

Order-preserving (OPE) OPE permite realizar cálculos de ordenamiento como ORDER BY, MIN, MAX, SORT, etc, ya que al tener dos textos planos, el uno mayor que el otro, los textos cifrados también conservarán su orden. Este esquema es menos seguro que DET porque revela orden.

Homomorphic encryption (HOM) Este esquema de encriptación es probabilístico y permite realizar cálculos computacionales sobre los datos encriptados. Para implementar cálculos de suma implementa un Criptosistema de Paillier [26]. Existe un tipo de encriptación totalmente homomórfica conocida como "Fully Homomorphic Encryption (FHE)" como por ejemplo la propuesta de Craig Gentry [27] el cuál se apoya en suposiciones criptográficas no suficientemente probadas y además son costosos computacionalmente [28] [27].

Join (JOIN y OPE-JOIN) Este esquema es necesario para permitir comparaciones de igualdad entre dos columnas, DET utiliza diferentes llaves para prevenir correlaciones cruzadas entre columnas. El esquema JOIN también soporta operaciones permitidas por DET y además permite determinar valores cifrados repetidos entre dos columnas. El esquema OPE-JOIN permite realizar uniones por relaciones de orden.

Word Search (SEARCH) El esquema SEARCH es usado para realizar búsquedas sobre texto cifrado y permite realizar operaciones como WHERE LIKE. En CryptDB se implementó un protocolo criptográfico descrito por Song et al. [29] para poder realizar este tipo de búsquedas, aunque se hicieron mejoras de seguridad.

2.2.3. Encriptación ajustable basada en las consultas

CryptDB fue creado con una parte fundamental para la seguridad y el dinamismo de las consultas SQL que se puedan realizar. Si la aplicación guarda un dato en una columna en la cuál no se realiza cálculos computacionales como ordenamiento o comparar valores, entonces se usará el esquema RND. Para las columnas que requieren comparación de valores se utilizará DET.

Uno de los problemas que puede ocurrir con este esquema, es que al momento de realizar consultas SQL las necesidades pueden ir cambiando, por ejemplo en un campo que utiliza el esquema RND puede requerir comparación de datos o obtener la suma de varios valores de esa columna. Es por esto que CryptDB implementa capas de encriptación conocida como *"Onion Encryption"*. Este mecanismo permite a CryptDB tener varias capas de encriptación aumentando la seguridad en cada capa.

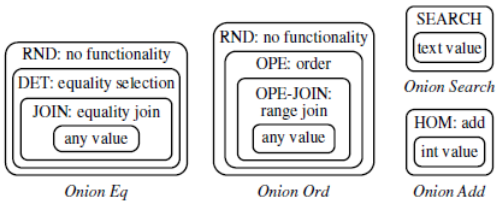


Figura 2.5: Onion Encryption

En la Figura 2.5 se puede evidenciar como CryptDB usa Onion Encryption para diferentes tipos de campos. Para cada capa, el proxy usa la misma llave para encriptar y desencriptar el valor de la misma columna pero utiliza una diferente a través de otras tablas, columnas, capas onion.

2.3. Escalamiento Horizontal y Vertical

Para poner en funcionamiento CryptDB, como ya se mencionó en secciones anteriores, se necesita un proxy de procesamiento de la información donde se encripta las consultas y se desencriptan los resultados. En el servidor donde va estar situada la aplicación y en el proxy vamos a aplicar técnicas

de escalamiento. Existen dos tipos de escalamientos: escalamiento vertical y escalamiento horizontal.

El escalamiento vertical consiste en elegir un servidor con mayores capacidades como CPU y/o memoria RAM. Este tipo de escalamiento es más costoso que un escalamiento horizontal. El escalamiento horizontal consiste en añadir un servidor auxiliar parecido al inicial, pero la carga se balancea entre los dos por el tiempo que sea necesario. Cuando la carga vuelva a la normalidad, se puede eliminar el servidor auxiliar Este tipo de escalamiento permite ahorrar costos mediante el incremento de servidores de manera temporal. Los servidores auxiliares se pueden incrementar cuando la carga computacional aumente.

2.4. Amazon Web Services(AWS)

AWS es la plataforma cloud que brinda múltiples servicios bajo demanda entre los que se destaca: renta de servidores (EC2), bases de datos(RDS), almacenamiento (S3), analítica, contenedores, etc. Estos servicios se pueden utilizar bajo demanda y el costo dependerá del tiempo en que se los use a los recursos.

A continuación, vamos a introducir los servicios AWS más relevantes.

2.4.1. AWS EC2

Es un servicio web que proporciona capacidad computacional en la nube segura y de tamaño modificable. Está diseñado para simplificar el uso de la informática en la nube a escala web para sus usuarios. En este servicio se puede rentar máquinas virtuales que se las denomina como “Instancias", las cuales ofrecen diferentes características como varios sistemas operativos, procesadores, memoria RAM y almacenamiento.

Amazon EC2 ofrece los procesadores rápidos de la nube y posee un ancho de banda de conexión con 400 Gbps de red de Ethernet. También posee procesadores GPU para la utilización en machine learning y las cargas de trabajo gráficas.

En este servicio se puede instalar servidores de aplicaciones web o aplicaciones en general y el costo que producirá dependerá del tiempo en que la instancia este siendo usada.

2.4.2. AWS Grupos de Auto escalamiento

Un grupo de auto escalamiento (Auto Scaling) contiene una colección de instancias Amazon EC2 que se tratan como una agrupación a efectos de escalado automático y administración. El tipo de escalamiento que se utiliza es el horizontal. Un grupo de Auto Scaling también permite utilizar características de Amazon EC2 Auto Scaling, como sustituciones de comprobaciones de estado y políticas de escalado. El mantenimiento del número de instancias en un grupo de Auto Scaling y el escalado automático son las funciones básicas del servicio de Amazon EC2 Auto Scaling.

Un grupo de Auto Scaling comienza lanzando suficientes instancias para satisfacer la capacidad deseada. Mantiene este número de instancias realizando comprobaciones de estado periódicas en las instancias del grupo. El grupo de Auto Scaling sigue manteniendo un número fijo de instancias aunque una de ellas deje de estar en buen estado. Si una instancia pasa a tener un estado incorrecto, el grupo termina la instancia en mal estado y lanza otra instancia para sustituirla. Se puede utilizar políticas de escalado para aumentar o disminuir dinámicamente el número de instancias del grupo para satisfacer las condiciones cambiantes. Cuando la política de escalado entra en vigor, el grupo de Auto Scaling ajusta la capacidad deseada del grupo, entre los valores de capacidad mínima y máxima, y lanza o termina las instancias según sea necesario.

2.4.3. RDS

Amazon RDS es un servicio administrado de base de datos relacional que pone a disposición seis motores de base de datos conocidos entre los que se puede elegir: Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server y PostgreSQL. Amazon RDS se encarga de las tareas habituales de las bases de datos, como

el aprovisionamiento, las revisiones, las copias de seguridad, la recuperación, la detección de errores y la reparación.

Con Amazon RDS se puede realizar la replicación de bases de datos para mejorar la disponibilidad y fiabilidad de las cargas de trabajo de producción. Con la opción de implementación Multi-AZ se permite ejecutar cargas de trabajo críticas con alta disponibilidad y conmutación por error automatizada e integrada desde la base de datos principal a una base de datos secundaria replicada sincrónicamente. Con las réplicas de lectura, puede escalar más allá de la capacidad de una implementación de una sola base de datos para cargas de trabajo de bases de datos con operaciones intensivas de lectura.

2.5. Load Balancers de AWS

Los Load Balancers en Amazon Web Services (AWS) son parte crucial en la arquitectura de aplicaciones escalables y resilientes. Un Load Balancer actúa como un distribuidor automático de tráfico entrante a través de múltiples destinos, como instancias de Amazon EC2, contenedores, y direcciones IP [30]

2.5.1. Tipos de Load Balancers

AWS ofrece tres tipos principales de Load Balancers, cada uno con características y casos de uso específicos:

- **Application Load Balancer (ALB):** Diseñado para el tráfico HTTP y HTTPS, el ALB opera en la capa 7 y ofrece una enrutación avanzada, incluyendo la capacidad de enrutar el tráfico basado en el contenido de la solicitud.

- **Network Load Balancer (NLB):** Trabaja en la capa 4 y es adecuado para manejar conexiones TCP y UDP. El NLB es conocido por su alta capacidad y baja latencia.

- **Classic Load Balancer (CLB):** Como el más antiguo de los tres, el CLB opera tanto en la capa 4 como en la capa 7, ofreciendo una solución básica de balanceo de carga.

2.5.2. Beneficios y Casos de Uso

Los Load Balancers de AWS ofrecen numerosos beneficios, incluyendo la distribución uniforme de cargas, la detección y el manejo de fallas en instancias, y la posibilidad de escalar automáticamente en función de la demanda. Son vitales en aplicaciones con alta disponibilidad y rendimiento.

2.6. Target Group

Un Target Group en AWS es una entidad clave que trabaja en conjunto con los Load Balancers. Un Target Group define dónde debe dirigirse el tráfico enrutado por el Load Balancer $\mathbf{AWSTargetGroup}$.

2.6.1. Funcionamiento y Configuración

Un Target Group consta de un grupo de recursos de destino (como instancias EC2 o funciones Lambda) y las reglas de enrutamiento para dirigir el tráfico hacia ellos. Puede ser configurado con protocolos específicos, puertos y rutas de salud para verificar la disponibilidad de los destinos.

2.6.2. Integración con Load Balancers

La integración entre Load Balancers y Target Groups permite una gestión eficiente y dinámica del tráfico. Los Load Balancers evalúan las reglas definidas en el Target Group y enrutan el tráfico de acuerdo a esas reglas, garantizando que las solicitudes se dirijan al recurso adecuado.

2.6.3. Consideraciones de Seguridad y Escalabilidad

Los Target Groups permiten la implementación de políticas de seguridad y autenticación, así como el manejo de sesiones pegajosas. Además, la integración con Auto Scaling Groups asegura que el Target Group pueda escalar según la demanda, ofreciendo una solución flexible y robusta para las aplicaciones modernas.

2.7. MIMIC III

Para poder realizar un estudio de rendimiento de la velocidad de respuesta de CryptDB que se acerque a un ambiente real, vamos a utilizar datos reales. Con este fin, hemos seleccionado la base de datos MIMIC para nuestro estudio.
MIMIC-III (Medical Information Mart for Intensive Care) es una gran base de datos que incluye información relacionada con los pacientes ingresados en las unidades de cuidados intensivos de un hospital. Los datos que constan en la base de datos son: signos vitales, mediciones de laboratorio, medicamentos, observaciones y notas registradas por los proveedores de atención, duración de la estadía en el hospital, códigos de procedimiento, códigos de diagnóstico, balance de líquidos, informes de imágenes, datos de supervivencia, entre otras. La base de datos está destinada para investigaciones de la academia e industria, iniciativas de mejora de la calidad y cursos de educación superior.
MIMIC-III contiene datos asociados con 53,423 ingresos hospitalarios distintos para pacientes adultos (de 16 años o más) ingresados en unidades de cuidados críticos entre 2001 y 2012. También, contiene datos de 7870 recién nacidos ingresados entre 2001 y 2008. Los datos cubren 38,597 distintos pacientes adultos y 49.785 ingresos hospitalarios. La mediana de edad de los pacientes adultos es de 65,8 años (Q1-Q3: 52,8-77,8), el 55,9 % de los pacientes son varones y la mortalidad hospitalaria es del 11,5 %. La duración media de una estancia en la UCI es de 2,1 días (Q1-Q3: 1,2-4,6) y la duración media de una estancia hospitalaria es de 6,9 días (Q1-Q3: 4,1-11,9). Se dispone de una media de 4579 observaciones registradas (“chartevents”) y 380 mediciones de laboratorio

(“labevents”) para cada ingreso hospitalario.

Antes de que los datos reales sean incorporados a la base de datos MIMIC-III, todos estos fueron anonimizados para dar cumplimiento con la Ley de Responsabilidad y Portabilidad del Seguro de Salud (HIPAA). El proceso de anonimización consta en remover dieciocho de los elementos entre los que se listan como por ejemplo: nombre del paciente, número telefónico, direcciones y fechas. Especialmente las fechas fueron cambiadas por fechas aleatorias en el futuro. Esta base de datos consta de 26 tablas relacionadas.

Capítulo 3

METODOLOGÍA

En el presente capítulo vamos a describir la metodología de investigación, la metodología de evaluación y las métricas de rendimiento que se utilizarán para llevar a cabo el presente proyecto.

3.1. Metodología de investigación

En el marco de la realización del presente proyecto, hemos adoptado una rigurosa metodología experimental descrita por Basili et al. [31], que se ajusta a los requisitos particulares de la investigación en ingeniería de software.
Esta metodología fue seleccionada tras un detenido análisis, debido a su enfoque en el proceso experimental y su aplicación exitosa en numerosos trabajos de investigación [32], [33], [34]. La metodología experimental de Basili consta de cuatro fases principales, cada una de ellas de vital importancia en el proceso de investigación:

1. **Definición**: Se lleva a cabo una revisión exhaustiva de la literatura existente y se articula de manera precisa la definición del problema. Además, se establecen los objetivos del estudio y se detallan las motivaciones. Para realizar la revisión de la literatura se utilizará la metodología propuesta por Kitchenham [35], la cual se detalle en la siguiente sección, Sección 3.2.

2. **Planificación**: Se delinea meticulosamente el diseño del estudio, se identifican y clasifican las variables, tanto independientes como dependientes, y se establecen los métodos de medición y evaluación que se aplicarán posteriormente.

3. **Ejecución u operación**: Esta etapa es el corazón del proceso experimental. Consiste en tres subfases: a) *Preparación*, en la que se ponen en marcha

todos los recursos necesarios; b) *Ejecución*, en la que se llevan a cabo los experimentos de acuerdo con los parámetros definidos; y c) *Análisis*, donde se evalúan y procesan los datos obtenidos.

4. **Interpretación**: La última fase es la interpretación de los resultados. Aquí se sintetizan los hallazgos, se les da sentido dentro del contexto del estudio, y se realiza un análisis profundo basado en cálculos estadísticos. Los resultados son extrapolados y comparados con estudios similares, y se evalúa su posible impacto en investigaciones futuras, así como en la práctica de la ingeniería de software en sí misma.

La estructura y coherencia inherentes a esta metodología experimental aseguran una implementación sólida y confiable, lo que convierte a este enfoque en una opción ideal para abordar los desafíos complejos y multidimensionales de la presente investigación.

3.2. Metodología de revisión sistemática de la literatura

Es crucial seleccionar una metodología adecuada que guíe la investigación. Existen múltiples metodologías ampliamente reconocidas para llevar a cabo revisiones sistemáticas, entre las que destacan PRISMA, diseñada principalmente para las ciencias médicas y de la salud [36]; Kitchenham, que se centra en la ingeniería de software [35]; y otros métodos como Cochrane y JBI que se usan en diferentes dominios[37] [38]. Cada metodología tiene sus propios méritos, deméritos y aplicaciones específicas. En el contexto de este proyecto de ingeniería de software, se ha optado por utilizar la metodología de Kitchenham por diversas razones que se explicarán en las siguientes subsecciones.

3.2.1. Comparación entre Metodologías: PRISMA vs. Kitchenham

Ambas metodologías, PRISMA y Kitchenham, proporcionan un marco estructurado para realizar revisiones sistemáticas. Sin embargo, existen diferencias clave

que hacen que una sea más adecuada que la otra en el contexto de la ingeniería de software.

3.2.1.1. PRISMA

PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) es ampliamente utilizado en el ámbito de las ciencias médicas. Aunque ofrece una estructura detallada para garantizar la calidad y la transparencia, está particularmente diseñado para revisiones que incluyen ensayos clínicos y estudios médicos. Su enfoque está en la meta-análisis y en criterios específicos que son más relevantes para las ciencias de la salud [36].

3.2.1.2. Kitchenham

Por otro lado, la metodología de Kitchenham se desarrolló específicamente para la ingeniería de software. Ofrece un marco flexible y adaptable que se ajusta bien a los variados tipos de investigaciones que se realizan en este campo. Los criterios de evaluación de la calidad y los pasos de la revisión son más apropiados para los estudios en ingeniería de software, desde estudios empíricos hasta revisiones teóricas [35].

3.2.1.3. Razones para Elegir Kitchenham

Optamos por utilizar la metodología de Kitchenham por las siguientes razones:

- **Relevancia al Campo:** Kitchenham se centra específicamente en la ingeniería de software, lo que lo hace más relevante para el presente trabajo.
- **Flexibilidad:** La metodología permite cierta flexibilidad para adaptarse a diferentes tipos de preguntas de investigación en ingeniería de software.
- **Profundidad de Análisis:** Kitchenham ofrece sub-pasos detallados que son esenciales para una evaluación en profundidad, desde la planificación hasta la documentación.

Aunque PRISMA es una metodología sólida para revisiones sistemáticas, su enfoque en las ciencias de la salud lo hace menos adecuado para el presente trabajo. La metodología de Kitchenham, en cambio, ofrece una estructura más apropiada y detallada para abordar las preguntas de investigación específicas de este campo.

3.2.2. Metodología Kitchenham

La metodología de revisión sistemática formulada por Barbara Kitchenham es un marco altamente estructurado para la síntesis de investigaciones previas, especialmente en el campo de la ingeniería de software. Este método se centra en tres fases fundamentales: Planificar la Revisión, Conducir la Revisión y Documentar la Revisión. Cada una de estas fases contiene varios sub-pasos específicos [35]. En la tabla 3.1 se puede ver un resumen de la metodología.

Tabla 3.1: Resumen de la Metodología de Revisión Sistemática de Barbara Kitchenham [35]

Fase	**Sub-fases**
Planificar la Revisión	Especificar Preguntas de Investigación Definir Criterios de Inclusión y Exclusión Definición de las Bases de Datos y Cadenas de Búsqueda
Conducir la Revisión	Identificar Fuentes/Estudios Relevantes Seleccionar Estudios Primarios Evaluar Calidad de los Estudios Extraer los Datos Requeridos Sintetizar Datos
Documentar la Revisión	Escribir Informe de Revisión Validar Informe

3.2.2.1. Planificar la Revisión

La primera fase es esencial para establecer el fundamento y el enfoque de la revisión. Los sub-pasos son los siguientes:

- **Especificar Preguntas de Investigación:** Las preguntas deben ser claras, precisas y relevantes para guiar todo el proceso de revisión.

- **Definir Criterios de Inclusión y Exclusión:** Se describen en detalle los criterios para determinar qué estudios deben ser incluidos o excluidos.
- **Definición de las Bases de Datos y Cadenas de Búsqueda:** Se seleccionan las bases de datos académicas donde se realizará la búsqueda, y se definen las cadenas de búsqueda o filtros de información.

3.2.2.2. Conducir la Revisión

Esta es la fase de ejecución, donde se realiza la búsqueda, selección y evaluación de los estudios. Los sub-pasos incluyen:

- **Identificar Fuentes/Estudios Relevantes:** Se emplean estrategias de búsqueda basadas en el protocolo de revisión.
- **Seleccionar Estudios Primarios:** Son los estudios que cumplen con los criterios establecidos en el protocolo de revisión.
- **Evaluar Calidad de los Estudios:** Se realiza una evaluación crítica de los estudios para asegurar la validez de la información.
- **Extraer los Datos Requeridos:** Se lleva a cabo la extracción de datos que son relevantes para las preguntas de investigación.
- **Sintetizar Datos:** Se interpretan los datos extraídos y se identifican patrones y tendencias.

3.2.2.3. Documentar la Revisión

La fase final se centra en garantizar que la revisión sea transparente y reproducible. Los sub-pasos son:

- **Escribir Informe de Revisión:** Se redacta un informe detallado, claro y conciso que proporciona una visión general de todo el proceso y hallazgos.

- **Validar Informe:** Se verifica que todos los aspectos del protocolo y de los resultados han sido debidamente cubiertos y documentados.

3.3. Metodología de evaluación

Para la realización del presente trabajo se adaptarán dos metodologías: la metodología para evaluación de aplicaciones de software propuesta por Cohill et al. [39] y la metodología para la evaluación del rendimiento de sistemas de almacenamiento de datos propuesta por Boral y DeWitt [40].

3.3.1. Aspectos de Evaluación

En base a las metodologías mencionadas, la evaluación en cuanto a software se realiza en relación a cinco aspectos: funcionalidad, usabilidad, rendimiento, soporte y documentación. A continuación, se describen brevemente estos aspectos:

Funcionalidad: Se refiere a lo que el producto puede hacer [39]. Incluye la métrica de cuantas tareas puede ejecutar un paquete de software, y se basa en listar y evaluar las tareas que el software necesita realizar [41].

Usabilidad: Medir la facilidad de uso de un software específico para el usuario final [39].

Rendimiento: Medir el costo utilizando un cronómetro, categorizado por tarea. Su objetivo principal es guiar el rendimiento esperado en un ambiente real [39].

Soporte: Medir la ayuda que se puede obtener del fabricante [39].

Documentación: Refiere a tutoriales, manuales, guías de implementación, con el objetivo de informar al usuario sobre cómo acceder y ejecutar funciones en el software.

Para los objetivos de este trabajo, se evaluarán los aspectos de documentación y rendimiento, enfocándose en el análisis y mejoramiento del rendimiento, además de evaluar la facilidad de instalación y uso de CryptDB usando la documentación disponible.

3.3.2. Evaluación de Documentación

La evaluación de documentación considerará seis parámetros según Plosh R. et al. [42], clasificados de 1 a 10, siendo 1 “Baja calidad” y 10 “Alta calidad”. Los parámetros son:

Precisión: Qué tan correcta es la documentación [42].

Claridad: Existencia de información clara sin ambigüedades [42].

Consistencia: Capacidad de prevenir conflictos [42].

Legibilidad: Facilidad con la que se puede leer [42].

Estructura: Nivel de organización de la información [42].

Facilidad de entendimiento: Facilidad con la que se puede entender [42].

Para medir los parámetros, se realizará un cuestionario que se puede encontrar en el Anexo B.2, respondiendo con “Sí” o “No” y una razón para la respuesta.

3.3.3. Evaluación de Rendimiento

La evaluación del rendimiento de bases de datos se realizará siguiendo a Boral et al. [40], considerando tres factores principales:

Nivel de multiprogramación: Refiere al número de tareas máximas que el motor de base de datos puede realizar en MySql [43].

Grado de compartición: Consultas múltiples que acceden a las mismas relaciones concurrentemente. Se mantendrá el mismo grado propuesto en [40].

Selección de consultas mezcladas: Selección de consultas representativas comunes en un ambiente real que podrían causar problemas de rendimiento [40].

3.4. Métricas de rendimiento

Las métricas de evaluación son un factor importante en el presente trabajo por lo cuál se utilizarán las siguientes métricas de rendimiento:

- **Latencia (milisegundos)**: Es el tiempo que toma entre la interacción del usuario y la respuesta del servidor.
- **Utilización del CPU (porcentaje):** Es la suma del trabajo manejado por la Unidad Central de Procesamiento (CPU).
- **Operaciones de Escritura por segundo (Número):** Son las operaciones de escritura completada por el unidad de almacenamiento.
- **Operaciones de Lectura por segundo (Número):** Son las operaciones de lectura completada por el unidad de almacenamiento.
- **Bytes de Escritura (Bytes):** Cantidad de bytes que están siendo escritos en la unidad de almacenamiento.
- **Bytes de Lectura (Bytes):** Cantidad de bytes que están siendo leídos en la unidad de almacenamiento.
- **Operaciones promedio Encoladas (Número):** Cantidad de operaciones encoladas promedio para la unidad de almacenamiento.

Estas métricas serán recolectadas del panel de monitoreo de AWS EC2 y AWS RDS en la sección de monitoreo.

3.5. Resumen de la metodología

Los pasos que se seguirán para llevar a cabo el presente proyecto están detallados en la Sección 3.1. En resumen, las fases se desarrollarán de la siguiente manera:

1. **Fase 1: Revisión de Literatura**
   En esta fase se seleccionarán los artículos científicos que constituyen la base del presente trabajo, y se añadirán investigaciones adicionales relacionadas con CryptDB, encriptación homomórfica, encriptación de bases de datos, AWS RDS, y API Rest.

2. **Fase 2: Definición de la Arquitectura**
   Aquí, se definirá la arquitectura de la solución, seleccionando específicamente las características de los servidores, el motor de base de datos, los datos que serán almacenados, el tipo de consultas que se realizarán para obtener las métricas, y el diseño de la API Rest.

3. **Fase 3: Metodología de Evaluación**
   Se seguirá la metodología descrita en la Metodología de Evaluación propuesta por Cohill et al. y Boral et al. [40] [39]. Como ya se mencionó, se evaluará solo el aspecto de rendimiento, tomando en cuenta las métricas descritas en la Sección 3.4.

4. **Fase 4: Recopilación de Datos**
   Se recogerán los datos obtenidos a través de herramientas como los tableros de monitoreo de AWS y los resultados de los tiempos de latencia de las consultas realizadas con K6.

5. **Fase 5: Documentación**
   Finalmente, los resultados obtenidos se documentarán en la sección de Análisis de resultados y en un artículo científico, consolidando las conclusiones y recomendaciones para futuras investigaciones.

Capítulo 4

EJECUCIÓN DEL PROYECTO

Para la ejecución del presente proyecto se va a seguir las diferentes fases antes expuestas en el Capítulo 3.

4.1. Fase 1 - Definición

La revisión sistemática de la literatura sobre CryptDB ha revelado una serie de estudios significativos centrados en el rendimiento y las optimizaciones de esta base de datos cifrada. Los artículos revisados abarcan desde aplicaciones específicas, como la optimización del rendimiento para la salud electrónica en la nube, hasta mejoras técnicas mediante el uso de aceleraciones específicas, como AES-NI y el teorema del residuo chino. Además, se han identificado desafíos clave, oportunidades futuras y pautas para el uso seguro de CryptDB. Estos trabajos proporcionan una comprensión integral de las capacidades y limitaciones de CryptDB, así como las direcciones futuras en el campo del cifrado de bases de datos. La revisión completa de estos estudios se puede encontrar en la sección 1.6, mientras que la metodología adoptada para esta revisión sistemática se describe detalladamente en la sección 3.2.

4.2. Fase 2 - Planificación

En esta fase se describirá e implementará los ambientes de pruebas y arquitecturas que serán utilizadas en el presente proyecto.

4.2.1. Ambientes de pruebas

Para poder hacer una comparativa del rendimiento de CryptDB, se utilizarán tres ambientes de pruebas. El primer ambiente contendrá los datos médicos sin cifrar a este lo llamaremos “Ambiente No Cifrado(ANC)”. El segundo ambiente contendrá los datos cifrados mediante CryptDB a este lo llamaremos “Ambiente

Cifrado Sin Escalamiento (ACSE)". Además de esto, para tratar de mejorar el rendimiento del ambiente ACSE se implementará un tercer ambiente el cuál lo llamaremos "Ambiente Cifrado Con Escalamiento(ACCE)".

4.2.2. Modelado Arquitectónico

4.2.2.1. Comparando Arquitecturas de Escalamiento Horizontal y Vertical

El escalamiento es una parte integral de las arquitecturas de software modernas, particularmente en la informática distribuida. La necesidad de utilizar eficientemente los recursos mientras se satisfacen demandas crecientes requiere dos tipos principales de escalamiento: horizontal y vertical.

Escalamiento Vertical El escalamiento vertical implica añadir más potencia a una máquina existente. En otras palabras, mejoras la capacidad de un solo servidor añadiendo más CPUs, memoria o almacenamiento [30].
Ventajas:

- *Simplicidad:* Más fácil de implementar ya que no hay necesidad de modificar el código de la aplicación.
- *Rendimiento:* Baja latencia debido a que todos los recursos se encuentran en un solo servidor.

Desventajas:

- *Costo:* El hardware de alta gama a menudo es costoso.
- *Límites de Escalabilidad:* Las restricciones físicas limitan cuánto se puede escalar.
- *Tiempo de Inactividad:* El escalamiento a menudo requiere tiempo de inactividad del sistema.

Escalamiento Horizontal El escalamiento horizontal, por otro lado, implica añadir más máquinas al grupo de recursos. El código de la aplicación puede

necesitar ser particionado para ejecutarse de manera distribuida [44] [30]. El escalamiento horizontal implica añadir más máquinas al sistema para distribuir la carga y mejorar el rendimiento. Para que este modelo funcione de manera eficiente, es crucial emplear un balanceador de carga. Este dispositivo o software distribuye las solicitudes entrantes de manera uniforme entre todos los servidores disponibles, optimizando así la utilización de recursos y minimizando los tiempos de espera.

Ventajas:

- *Alta Disponibilidad:* Los sistemas pueden seguir funcionando de manera efectiva incluso si uno o más servidores fallan.
- *Elasticidad:* Más fácil adaptarse a cambios en la demanda [30].
- *Sin Límites Físicos:* Más fácil añadir más máquinas a medida que crecen tus requisitos.

Desventajas:

- *Complejidad:* El código de la aplicación necesita ser diseñado para escalar.
- *Latencia de Red:* Un potencial para una mayor latencia debido a la distribución de datos e inclución de un balanceador de carga.

Cuándo Elegir Qué El escalamiento vertical suele ser una buena elección para aplicaciones con un conjunto de datos más pequeño o aquellas que requieren acceso frecuente a los datos [45]. El escalamiento horizontal brilla cuando tienes un conjunto de datos grande y distribuido o cuando la alta disponibilidad y tolerancia a fallos son cruciales.

Las mejores prácticas de AWS recomiendan el escalamiento horizontal para sistemas distribuidos para satisfacer las demandas de la computación en la nube [30]. Aunque el escalamiento vertical puede ofrecer una solución rápida para problemas de rendimiento, el futuro se inclina indiscutiblemente hacia el escalamiento horizontal, ya que ofrece más espacio para el crecimiento y la adaptabilidad.

4.2.2.2. Arquitectura propuesta

Tanto el escalamiento vertical como el horizontal tienen sus méritos y desventajas. Sin embargo, a medida que los conjuntos de datos crecen exponencialmente y la informática distribuida se convierte en la norma, las ventajas del escalamiento horizontal son demasiado numerosas para pasarlas por alto.

La arquitectura con escalamiento horizontal que es candidata a considerarse en el presente proyecto consta de seis servidores base alojados en la nube, Figura 4.1.

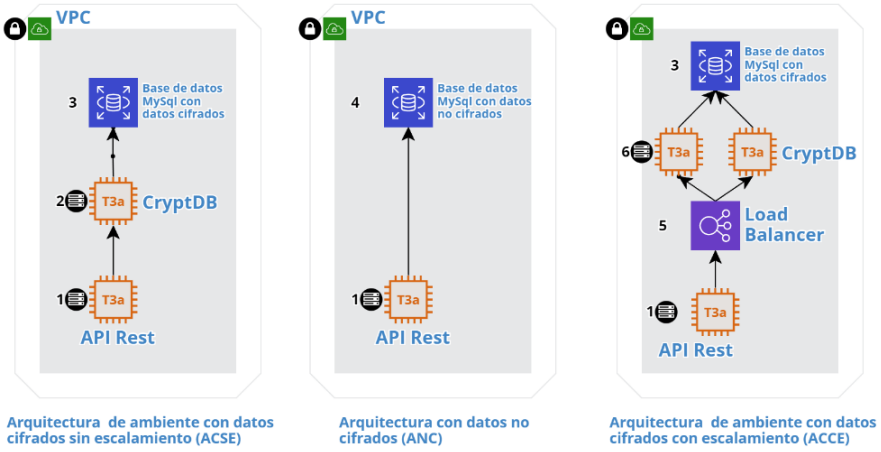


Figura 4.1: Arquitectura de la solución

En la Figura 4.1 se muestra la arquitectura de los tres ambientes de evaluación: ACSE, ANC y ACCE. Cada servidor, sea de base de datos, de aplicación o balanceador de carga está etiquetado con un número del 1 al 6.

El servidor 1 que se encuentra presente en las 3 arquitecturas, contiene el API que consulta la información de la base de datos. Este servidor es de tipo t3a. Este tipo t3a y t3 se refiere a un servidor de uso general como lo define AWS. En el servidor 2, que se encuentra presente en la arquitectura ACSE instalaremos y configuraremos CryptDB, este servidor es de tipo t3a. Los servidores 3 y 4, presentes en las 3 arquitecturas, ofrecen el servicio de database-as-a-service, DaaS, conocido como AWS RDS de tipo t3. Los servidores 3 y 4 almacenarán, respectivamente, los datos cifrados y no cifrados. Todos los servidores tendrán un almacenamiento tipo SSD (Solid State Disk) de uso general. El número 5 presente en la arquitectura ACCE, es el servicio de balanceo de carga de AWS conocido como “Elastic Load Balancer (ELB)” el cual va a apuntar a los servidores que

contienen a CryptDB. El número 6 presente en la arquitectura ACCE, representa el cluster (grupo de servidores) que contiene a CryptDB y es usado para distribuir la carga de las consultas a los servidores que tienen CryptDB.
En AWS, los servidores virtuales se les conoce como instancias. Se instala CryptDB sobre instancias de AWS EC2 para conexiones al servicio de base de datos. Esta contiene los registros médicos cifrados y otra instancia de MySQL contendrá los datos no cifrados. El API Rest se sometió a pruebas de carga comparativas entre las bases de datos cifrada y no cifrada. Las pruebas fueron divididas en lectura, escritura y actualización de datos. Se realiza un análisis de rendimiento de tiempo de respuesta por medio de la latencia. El objetivo es determinar problemas de CPU, disco de almacenamiento o memoria RAM en algún punto de la arquitectura.
El presente proyecto busca verificar si existe un cuello de botella en cualquiera de los componentes de la arquitectura. Para el análisis se comparará las arquitecturas con datos cifrados y datos sin cifrar. Finalmente se sugerirá posibles soluciones a los problemas encontrados.

4.2.3. Gestor de base de datos

En el trabajo presentado en [3], se utiliza MySql como gestor de base de datos. Para poder alinear el presente trabajo con el de Faisal Shahzad et al, se utilizará el mismo gestor de base de datos. Se tenía planeado utilizar el servicio ofrecido por AWS llamado Amazon RDS(Relational Database System), pero por un problema de compatibilidad con MySQL y CryptDB se decidió instalar Mysql en una instancia EC2. El problema de compatibilidad es por la razon que CryptDB máximo soporta Mysql 5.5 y AWS RDS provee mínimo la versión 5.7.33.

4.2.4. Base de datos

Como ya se ha discutido en secciones anteriores, se utilizará MIMIC III(Medical Information Mart for Intensive Care III) como base de datos de registros médicos. Estos registros de base de datos son anonimizados y con fechas adelantadas al futuro, pero con datos reales de personas que estuvieron internados en

UCI(Unidad de Cuidados Intensivos). Esta base de datos es de libre acceso, pero requiere un curso para poder acceder a los datos y poder descargarlos.
La base de datos consta de 26 tablas relacionadas, los datos se encuentran en formato CSV. Para poder migrarlos y almacenarlos en MySQL, se utiliza una herramienta proveida por MIMIC III en su repositorio de github llamado "buildmimic". La herramienta consta de 3 scripts que leen los archivos CSV y almacena en la base de datos predefinida en los scripts.
Las tablas están entrelazadas por identificadores que generalmente tienen el sufijo "ID". Por ejemplo, HADM_ID se refiere a un ingreso hospitalario único y SUBJECT_ID se refiere a un paciente único. Una excepción es ROW_ID, que es simplemente un identificador único de fila para esa tabla [46]. Las tablas prefijadas con "D_" son diccionarios y proporcionan definiciones para identificadores. Por ejemplo, cada fila de OUTPUTEVENTS está asociada con un único ITEMID que representa el concepto medido, pero no contiene el nombre real del fármaco. Al unir OUTPUTEVENTS y D_ITEMS por medio de ITEMID, es posible identificar qué concepto representa un ITEMID dado [46].
Las tablas que se pueden encontrar en MIMIC III son las siguientes:

- **ADMISSIONS**: Cada hospitalización única para cada paciente en la base de datos (define HADM_ID) [46].
- **CALLOUT**: Información sobre cuándo un paciente fue autorizado para el alta de en UCI y cuándo fue realmente dado de alta [46].
- **ICUSTAYS**: Cada permanencia única en UCI en la base de datos (define ICUSTAY_ID)[46].
- **PATIENTS**: Cada paciente único en la base de datos (define SUBJECT_-ID)[46].
- **SERVICES**: El servicio clínico bajo el cual está registrado un paciente [46].
- **TRANSFERS**: Transferencia del paciente de una cama a otra dentro del hospital, incluida la admisión y el alta de la UCI [46].

Las siguientes tablas contienen datos recopilados en la unidad de cuidados intensivos:

- **CAREGIVERS**: Todo cuidador(doctor o enfermero) que haya registrado datos en la base de datos (define CGID).
- **CHARTEVENTS**: Todas las observaciones registradas de los pacientes.
- **DATETIMEEVENTS**: Todas las observaciones registradas las cuales son fechas, por ejemplo, hora de diálisis o inserción de líneas.
- **INPUTEVENTS_CV**: Ingesta para pacientes monitoreados usando el sistema Philips CareVue mientras están en la UCI.
- **INPUTEVENTS_MV**: Ingesta para pacientes monitoreados usando el sistema iMDSoft Metavision mientras están en la UCI.
- **NOTEEVENTS**: notas anonimizadas, incluidas notas de enfermería y de médicos, informes de ECG(Electrodiagram), informes de imágenes y resúmenes de alta.
- **OUTPUTEVENTS**: información de salida para los pacientes mientras están en la UCI.
- **PROCEDUREEVENTS_MV**: Procedimientos en pacientes para el subconjunto de pacientes que fueron monitoreados en la UCI mediante el sistema iMDSoft MetaVision.

Las siguientes tablas contienen datos recopilados en la unidad de cuidados intensivos:

- **CPTEVENTS**: Procedimientos registrados como códigos de Terminología de procedimiento actual (CPT).
- **DIAGNOSES_ICD**: Diagnósticos asignados por el hospital, codificados utilizando el sistema de Clasificación Estadística Internacional de Enfermedades y Problemas de Salud Relacionados (CIE).

- **DRGCODES**: Grupos relacionados con el diagnóstico (DRG), que son utilizados por el hospital para fines de facturación.

- **LABEVENTS**: Mediciones de laboratorio para pacientes tanto dentro del hospital como en consultas externas.

- **MICROBIOLOGYEVENTS**: Mediciones microbiológicas y sensibilidades de la base de datos del hospital.

- **PRESCRIPTIONS**: Medicamentos ordenados, y no necesariamente administrados, para un paciente determinado.

- **PROCEDURES_ICD**: Procedimientos en el paciente, codificados utilizando el sistema de Clasificación Estadística Internacional de Enfermedades y Problemas de Salud Relacionados (ICD).

Las siguientes tablas son diccionarios:

- **D_CPT**: Diccionario de alto nivel de códigos de terminología de procedimiento actual (CPT).

- **D_ICD_DIAGNOSES**: Diccionario de códigos de la Clasificación Estadística Internacional de Enfermedades y Problemas Relacionados con la Salud (CIE) relacionados con los diagnósticos.

- **D_ICD_PROCEDURES**: Diccionario de códigos de la Clasificación Estadística Internacional de Enfermedades y Problemas Relacionados con la Salud (CIE) relacionados con los procedimientos.

- **D_ITEMS**: Diccionario de ITEMIDs que aparecen en la base de datos MIMIC, excepto los que se refieren a pruebas de laboratorio.

- **D_LABITEMS**: Diccionario de ITEMID en la base de datos del laboratorio que se relaciona con las pruebas de laboratorio.

4.2.5. Elaboración de Consultas a la Base de Datos

Las consultas a la base de datos se utilizarán para poder medir el rendimiento de CryptDB. Estas se pondrán en la aplicación API que realizará las consultas al motor de Base de Datos MySQL usando CryptDB.
Para selección y elaboración de consultas SQL realizadas a la Base de Datos, los parámetros de selección se basan en lo que propuso Boral et al. [40]. Se debe seleccionar consultas mezcladas representativas que representen un ambiente real y podrían causar problemas de rendimiento. Además se debe evaluar el grado de compartición de los datos entre las consultas.
Para entender que datos son importantes consultar sobre pacientes en UCI, se consultó un Médico con experiencia en UCI. La encuesta se puede ver en los Anexo B1. Como resultado de la encuesta se puede evidenciar que las consultas más frecuentes son las siguientes:

- Hoja de evolución por hora, esto integra por ejemplo: signos vitales, presión arterial, frecuencia respiratoria, glucosa, entre otros.
- Resultados de Laboratorio que se realizan al menos una o dos veces al día
- La medicación que se está administrando y la frecuencia de esta.

Las tablas que integran MIMIC III y contienen los datos requeridos en el mismo orden de los resultados de la encuesta son los siguientes:

- CHARTEVENTS contiene toda la información respecto a las observaciones de los pacientes, como por ejemplo: PH Arterial, Glocusa, Hematocritos, CO2 Arterial, Calcio Ionizado, entre otros.
- LABEVENTS contiene toda la información respecto a los exámenes de laboratorios de los realizados a pacientes, como por ejemplo: Saturación de Oxígeno, CO2 Calculado, Nivel de Magnesio, Conteo de Plaquetas, entre otros.

- PRESCRIPTIONS contiene toda la información respecto a los medicamentos ordenados para los pacientes, como por ejemplo: Insulina, Nitroglicerina, Metropolol, entre otros.

En base a esto, es necesario realizar tres consultas que obtengan los datos necesarios para que un médico sepa la situación del paciente. Para obtener las observaciones de los pacientes con las etiquetas correctas es necesario realizar una unión de la tabla “CHARTEVENTS” con la tabla “D_ITEMS”. En la Figura A.1 del Anexo A se puede ver las relaciones con otras Tablas.
Para obtener los resultados de los exámenes de laboratorio realizados a los pacientes con las etiquetas correctas es necesario realizar una unión de la tabla “LABEVENTS” con la tabla “D_LABITEMS”. En la Figura A.2 del Anexo A se puede ver las relaciones con otras Tablas.
Para obtener las prescripciones de los medicamentos ordenados para los pacientes se realiza una consulta a la tabla “PRESCRIPTIONS”. En la Figura A.3 del Anexo A se puede ver las relaciones con otras Tablas.
Como resultado se puede ver las consultas SQL de la siguiente manera en el orden correspondiente:

```
SELECT  it.LABEL as 'Item', ce.VALUE, ce.VALUEUOM,
ce.STORETIME as 'Fecha y Hora'
FROM mimiciiiv14.CHARTEVENTS as ce
join mimiciiiv14.D_ITEMS it on ce.ITEMID=it.ITEMID
where SUBJECT_ID=<PARAM>
order by ce.STORETIME desc;

SELECT  it.LABEL as 'Item', le.VALUE, le.VALUEUOM,
le.CHARTTIME as 'Fecha y Hora'
FROM mimiciiiv14.LABEVENTS as le
join mimiciiiv14.D_LABITEMS it on le.ITEMID=it.ITEMID
where SUBJECT_ID=<PARAM>
order by le.CHARTTIME desc;
```

```
SELECT  *
FROM mimiciiiv14.PRESCRIPTIONS as pr
where SUBJECT_ID=<PARAM>;
```

4.2.6. Diseño del API Rest

Python con Flask es una lenguaje y framework de desarrollo de software de código abierto y gratuito para crear muchos tipos de aplicaciones, como: aplicaciones web, API web y microservicios, funciones sin servidor en la nube, aplicaciones nativas de la nube, aplicaciones móviles, aplicaciones de escritorio entre otros.
El API Rest se lo realizará con tecnología Flask con el lenguaje de programación Python. Esta versión permite realizar aplicaciones multiplataforma como por ejemplo: Windows, Linux, MacOS, Android, iOS entre otros. La estructura de la aplicación será de WebAPI, por eso se seleccionó Flask como framework de desarrollo. Flask es una tecnología que es mantenida por una comunidad global con el soporte principal de la comunidad Flask de Python [47].
El API Rest estará compuesto de tres capas. La primera capa son los Controladores, los cuales son el primer punto de entrada de una petición y son los encargados de responder la petición. Esta capa también contendrá cualquier lógica de negocio como validaciones, procesos y control de transacciones. La segunda capa es la de Repositorios, aquí estarán localizados las consultas SQL hechas a las bases de datos como de obtención, actualización, creación y borrado de datos. La tercera capa son los Modelos, estos son la representación de las tablas de la base de datos, peticiones y respuesta del API Rest. En la Figura 4.2, se puede ver de manera gráfico las capas de la arquitectura de la aplicación.

4.3. Fase 3 - Ejecución y Resultados

En esta sección se describe los pasos que fueron utilizados para la ejecución, experimentación y análisis del presente trabajo. En el repositorio del proyecto

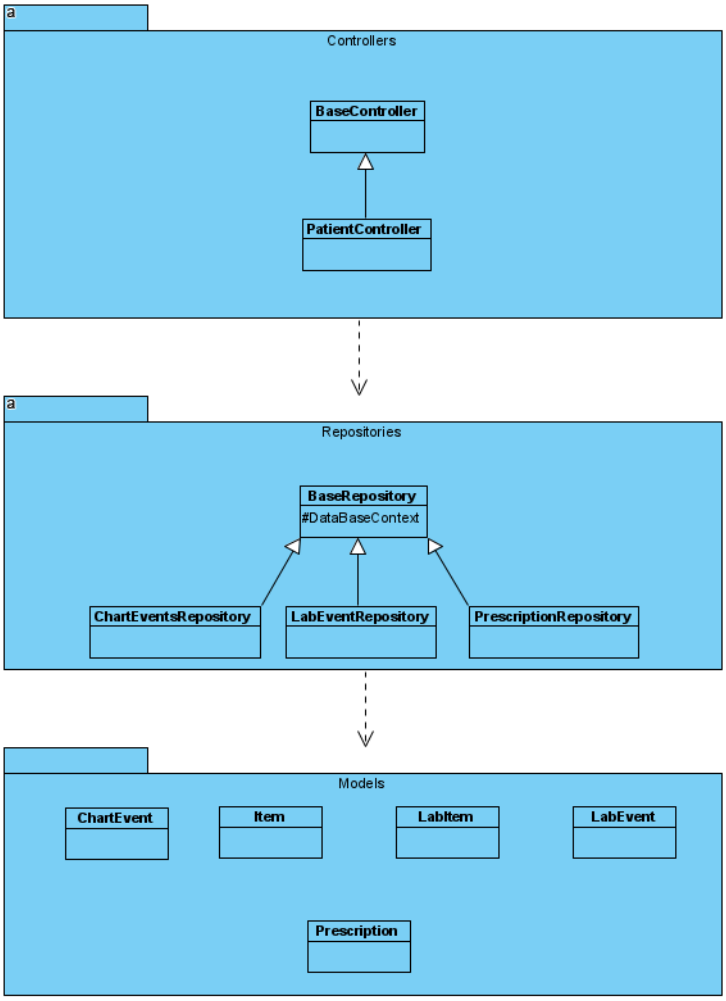


Figura 4.2: Diagrama de arquitectura de la aplicación

https://github.com/sgarciab/ThesisCryptDBLoadTesting se puede encontrar los recursos que se mencionan en esta sección. Las siguientes secciones se refieren a este repositorio como el “repositorio origen”.

4.3.1. Preparación

4.3.1.1. Obtención y Preparación de la Base de Datos MIMIC III

MIMIC III En este proyecto, enfrentamos el desafío monumental de acceder, transformar, y configurar las bases de datos III. A continuación, se detalla este laborioso viaje lleno de obstáculos, aprendizajes, y éxitos.

Etapa 1: Acceso a III

CITI Program's Course La obtención de acceso a MIMIC III requirió inicialmente completar el curso CITI Program's "Data or Specimens Only Research". La aprobación de este curso permitió el poder continuar con el trabajo.

Descarga y Transformación Los datos descargados en formato CSV tuvieron que ser meticulosamente convertidos y estructurados. El proceso de conversión fue laborioso, nos llevó aproximadamente dos meses, ya que la base de datos se corrompía repetidamente. Finalmente, después de cinco intentos fallidos, logramos crear un script exitoso para insertar los datos en MySQL y luego a CryptDB.

Etapa 2: Configuración y Pruebas de Carga

Creación de Scripts Se creó un script en Python para popular las tablas de MySQL, un proceso que también fue una tarea cuidadosa al momento de evaluar el contenido de las tablas correctas. Este script se encuentra disponible en el repositorio origen, en la carpeta scripts/import_csv_to_mysql.

Incompatibilidad con AWS Nos enfrentamos a la incompatibilidad de CryptDB con AWS RDS. CryptDB requería MySQL versión 5.1, mientras que AWS solo ofrecía las versiones 5.7 y 8.0. Este obstáculo inesperado requirió un abordaje diferente.

Etapa 3: Instalación en AWS EC2

Selección de Ubuntu Después de una investigación, optamos por Ubuntu 14.04.6 para instalar CryptDB y MySQL en un servidor de AWS EC2. Este sistema operativo fue seleccionado debido a la falta de repositorios de CryptDB compatibles con versiones más recientes.

Uso de Repositorio GitHub Se encontró un repositorio en GitHub, https://github.com/yiwenshao/Practical-Cryptdb, que permitió la instalación en una versión de Ubuntu compatible con AWS EC2. A pesar de su compatibilidad,

la configuración de este repositorio presentó su propio conjunto de desafíos y aprendizajes.

Configuración de Instancias EC2 y Load Balancer para CryptDB La integración de CryptDB con AWS fue llevada a cabo para asegurar la distribución eficiente de la carga y la escalabilidad. En este escenario, se configuraron instancias EC2, junto con un Load Balancer para balancear las cargas a CryptDB, mientras se manejaba el tráfico desde una API instalada en una EC2. A continuación, se describen los pasos que se siguieron.

Creación de Instancias EC2 para CryptDB

1. **Lanzamiento de la Instancia EC2:** Se inició una nueva instancia EC2 desde la consola de AWS, seleccionando la imagen de máquina adecuada para CryptDB.
2. **Configuración de Seguridad:** Se establecieron reglas de grupo de seguridad para permitir el tráfico necesario para CryptDB y la API.
3. **Instalación de CryptDB:** Se conectó a la instancia EC2 y se procedió a instalar CryptDB siguiendo las instrucciones del repositorio.

Creación de Instancias EC2 para la API

1. **Lanzamiento de la Instancia EC2:** Se inició una nueva instancia EC2 para la API.
2. **Configuración de Comunicación:** Se aseguró que la API pudiera comunicarse con CryptDB, ajustando las reglas de seguridad según fue necesario.
3. **Instalación de la API:** Se instaló y configuró la API en esta instancia EC2.

Configuración del Load Balancer

1. **Creación del Load Balancer:** Desde la consola de AWS, se creó un Network Load Balancer según las necesidades.

2. **Configuración del Target Group:** Se creó un Target Group, añadiendo las instancias EC2 de CryptDB como destinos.
3. **Configuración de Reglas de Enrutamiento:** Se establecieron las reglas de enrutamiento para dirigir el tráfico de la API hacia CryptDB a través del Load Balancer.
4. **Verificación de Salud:** Se configuró la verificación de salud para asegurar que el tráfico solo fuera dirigido a instancias saludables de CryptDB.

Esta configuración permitió un manejo eficiente y escalable del tráfico entre la API y CryptDB, aprovechando las capacidades de balanceo de carga de AWS. La integración de estas tecnologías aseguró un alto rendimiento y disponibilidad, alineando la infraestructura con las necesidades de la aplicación moderna.

Etapa 4: Creación del Script de Prueba de Carga

Uso de la Librería k6 Desarrollamos un script para realizar la prueba de carga, utilizando la librería k6, una herramienta que permite escribir comandos en JavaScript para simular múltiples conexiones concurrentes. El script, alojado en el repositorio origen bajo el nombre de `script.js`, llama al URL del API Rest que responde con los datos de los pacientes.

Etapa 5: Construcción y Migración del API Rest

Inicialización en .Net Inicialmente, el API Rest fue construido en .Net, pero nos enfrentamos a la incompatibilidad del driver de conexión con MySQL para CryptDB. Esta incompatibilidad nos llevó a buscar alternativas.

Migración a Python y Flask Finalmente, decidimos migrar el API Rest a Python, aprovechando Flask, un framework que facilita la creación de API Rest. El código fuente del API, disponible en el repositorio origen como `api.py`, refleja una solución exitosa a los problemas de compatibilidad.

Etapa 7: Configuración del Entorno de Pruebas

Ubicación de los Servidores K6 y el API Rest fueron configurados en un servidor Ubuntu 20.04, ubicado en la misma subred de la VPC de AWS. Los servidores de CryptDB y MySQL también se encuentran en esta subred.

Minimización de la Latencia de conexión entre servidores Gracias a la disposición de los servidores, logramos una latencia entre conexiones de un máximo de 0.16 milisegundos y un promedio de 0.09 milisegundos [48]. Esta configuración óptima garantiza una comunicación eficiente y rápida entre los diferentes componentes del sistema.

4.3.2. Ejecución y Resultados

La fase de ejecución está integrada por pruebas de carga, con el objetivo de encontrar posibles cuellos de botella en la arquitectura ACSE, ANC y ACCE.
Al ejecutar las pruebas de carga, se configura los siguientes parámetros en la herramienta k6:

- Número de usuarios concurrentes: 10, 50, 100
- Duración de la prueba: 10 minutos
- Parada cuidadosa(Graceful Stop): 30 segundos

Este último parámetro, parada cuidadosa, da un tiempo donde las peticiones que aún no han respondido desde el API Rest tengan el tiempo de máximo 30 segundos para responder y no marcar esas peticiones como no resueltas. Este es un ejemplo del comando que se ejecuta:

```
k6 run --vus 50 --duration 600s
--out csv=k6_cryptdb0_node_1_users_50.csv script.js
```

El parámetro –vus se refiere a los usuarios concurrentes y el parámetro –duration es la cantidad de segundos que va correr la prueba. Finalmente, el parámetro –out indica en que archivo se van a guardar los resultados.
Además de los resultados a detalle, k6 brinda un resumen que contiene las estadísticas de los resultados de la prueba de carga. Como por ejemplo en la

Figura 4.3, se puede ver los parámetros principales de medición que se obtiene como resultado.

```
k6 run --vus 10 --duration 600s --out csv=k6_cryptdb0_node_1_users_10.csv script.js
 data_received..................: 7.4 MB 12 kB/s
 data_sent......................: 187 kB 298 B/s
 http_req_blocked...............: avg=21.42µs min=1.44µs  med=3.67µs   max=9.34ms  p(90)=4.98µs  p(95)=5.81µs
 http_req_connecting............: avg=16.44µs min=0s      med=0s       max=9.26ms  p(90)=0s      p(95)=0s
 http_req_duration..............: avg=3.61s   min=98.91ms med=280.32ms max=57.66s  p(90)=10.54s  p(95)=27.89s
   { expected_response:true }...: avg=3.61s   min=98.91ms med=280.32ms max=57.66s  p(90)=10.54s  p(95)=27.89s
 http_req_failed................: 0.00%  ✓ 0         ✗ 1740
 http_req_receiving.............: avg=14.38ms min=53.7µs  med=290.06µs max=59.39ms p(90)=43.53ms p(95)=44.91ms
 http_req_sending...............: avg=27.91µs min=10.08µs med=19.09µs  max=1.51ms  p(90)=30.65µs p(95)=39.94µs
 http_req_tls_handshaking.......: avg=0s      min=0s      med=0s       max=0s      p(90)=0s      p(95)=0s
 http_req_waiting...............: avg=3.6s    min=98.74ms med=261.82ms max=57.66s  p(90)=10.54s  p(95)=27.89s
 http_reqs......................: 1740   2.761889/s
 vus............................: 10     min=10     max=10
 vus_max........................: 10     min=10     max=10
```

Figura 4.3: Ejemplo de resumen de k6

Para las pruebas de carga se tiene 6 escenarios a ejecutarse:

- 10 usuarios concurrentes con 1 nodo de CryptDB
- 10 usuarios concurrentes con 2 nodos de CryptDB
- 50 usuarios concurrentes con 1 nodo de CryptDB
- 50 usuarios concurrentes con 2 nodos de CryptDB
- 100 usuarios concurrentes con 1 nodo de CryptDB
- 100 usuarios concurrentes con 2 nodos de CryptDB

4.3.3. Análisis de Resultados

En esta subsección se podrá ver los resultados del análisis de la documentación de CryptDB y de las pruebas de carga ejecutadas con todas las arquitecturas.

4.3.4. Documentación

En esta sección explicamos los resultados obtenidos para el aspecto de documentación, con los parámetros de calidad.

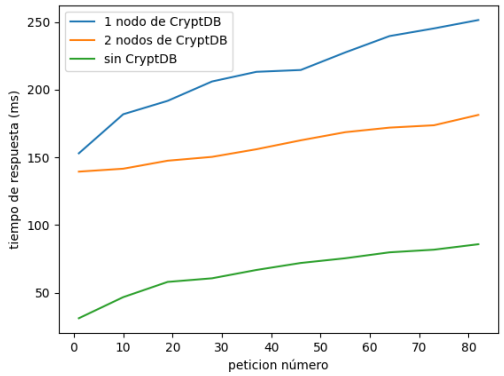
En la sección del Anexo B.2 se puede ver el cuestionario mencionado en la metodología de la evaluación, ya con las respuestas a las preguntas sobre la calidad de documentación.

El calificación final es 2 sobre 11, la cual denota que la calidad de la documentación es mala. Por lo tanto si algún proyecto o empresa quisieran implementarla en su conjunto tecnológico, tomaría mucho tiempo el poder poner a andar el proyecto de CryptDB y integrarlo con herramientas ya existentes.

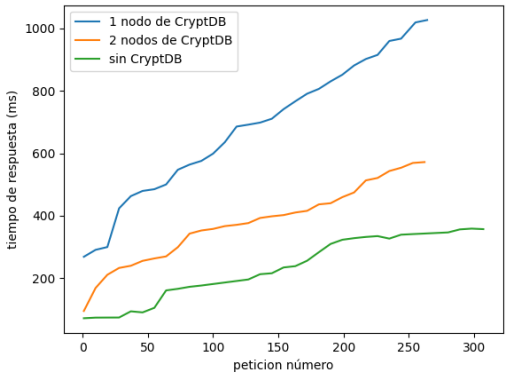
4.3.5. Rendimiento

En esta sección, explicamos los resultados obtenidos para el aspecto rendimiento con la prueba de carga con los distintos usuarios.

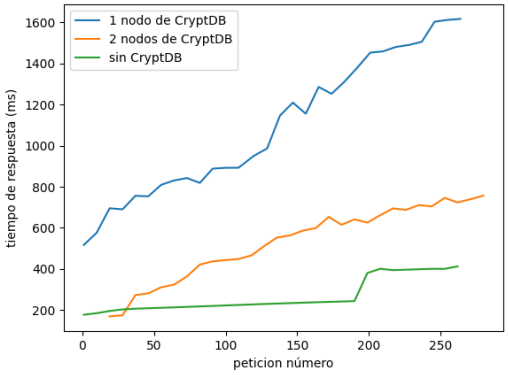
4.3.5.1. Latencia



(a) 10 usuarios concurrentes



(b) 50 usuarios concurrentes



(c) 100 usuarios concurrentes

Figura 4.4: Concurrencia

En las Figuras 4.4a 4.4b 4.4c se puede ver los tiempos de respuesta conforme el tiempo va pasando y el API responde los datos solicitados. En la leyenda

de las figuras se encuentra 3 elementos: “1 nodo de CryptDB”, “2 nodos de CryptDB”, “sin CryptDB”. El primero “1 nodo de CryptDB” indica y hace referencia a la Figura 4.1 “Arquitectura de ambiente con datos cifrados sin escalamiento (ACSE)”. El segundo “2 nodos de CryptDB” indica y hace referencia a la Figura 4.1 “Arquitectura de ambiente con datos cifrados con escalamiento(ACCE)”. Y el tercero “sin CryptDB” indica y hace referencia a la Figura 4.1 “Arquitectura con datos no cifrados(ANC)”

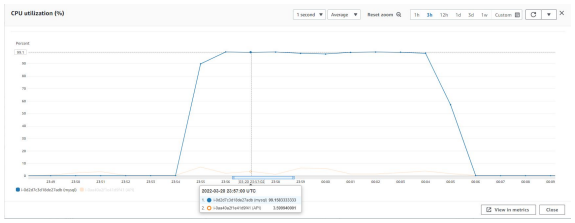
La Figura 4.4a muestra a 10 usuarios concurrentes, haciendo consultas durante 10 minutos. Los tiempos de respuesta se van incrementando con forme el tiempo pasa. En este ejemplo se puede ver claramente que tener 2 nodos de CryptDB(ACCE) ayuda a mejorar los tiempos de respuesta en comparación a tener 1 solo nodo(ACSE). Pero no se acerca al mismo rendimiento que tiene una comunicación directa sin CryptDB(ANC).

La Figura 4.4b y la Figura 4.4c muestran a 50 usuarios y a 100 concurrentes respectivamente, haciendo consultas durante 10 minutos. El comportamiento es parecido al de la Figura 4.4a pero aquí se puede evidenciar una mejora significativa que no se lograba apreciar con 10 usuarios concurrentes. La mejora puede ser de hasta un 50 % en el tiempo de respuesta. Claramente se puede ver que tener un solo nodo con CryptDB es un cuello de botella en cuanto a lectura. Cabe recalcar que las pruebas de escritura y actualización no se pudieron realizar ya que al momento de realizar escritura o actualización con múltiples nodos de CryptDB, se genera un error de integridad de datos por lo cual no es posible tener múltiples nodos de CryptDB.

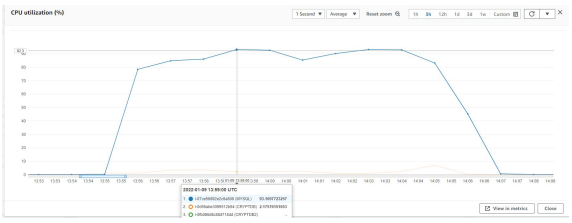
4.3.5.2. Utilización del CPU

Las pruebas con 10, 50 y 100 usuarios que incluyen y excluyen a CryptDB demuestran que la utilización del CPU de CryptDB es mínima en todos los casos en comparación a la de Mysql . En las Figuras 4.5a 4.5b 4.5c 4.5d 4.5e 4.5f se evidencia que el CPU de la instancia de CryptDB no es un problema.

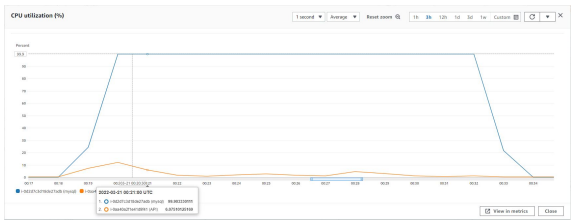
Hasta ahora esto indica que CryptDB no está diseñado para alta concurrencia y es el principal problema para los altos tiempos de respuesta.



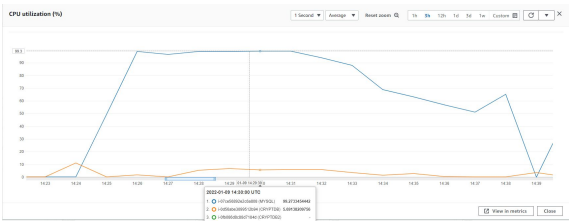
(a) Uso del CPU con 10 usuarios concurrentes sin CryptDB



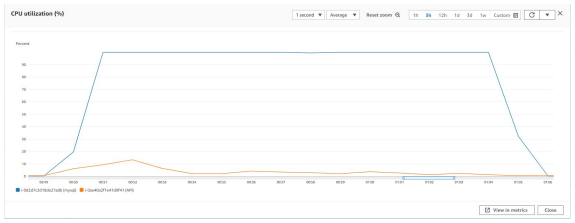
(b) Uso del CPU con 10 usuarios concurrentes con CryptDB



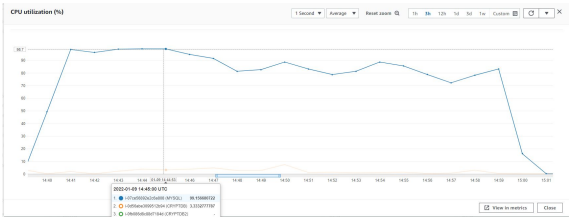
(c) Uso del CPU con 50 usuarios concurrentes sin CryptDB



(d) Uso del CPU con 50 usuarios concurrentes con CryptDB



(e) Uso del CPU con 100 usuarios concurrentes sin CryptDB



(f) Uso del CPU con 100 usuarios concurrentes

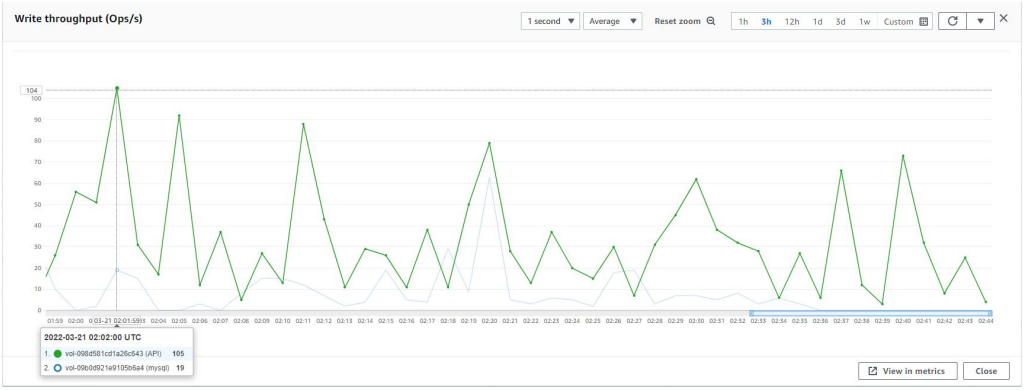
Figura 4.5: Utilización del CPU

4.3.5.3. Operaciones de Escritura por segundo

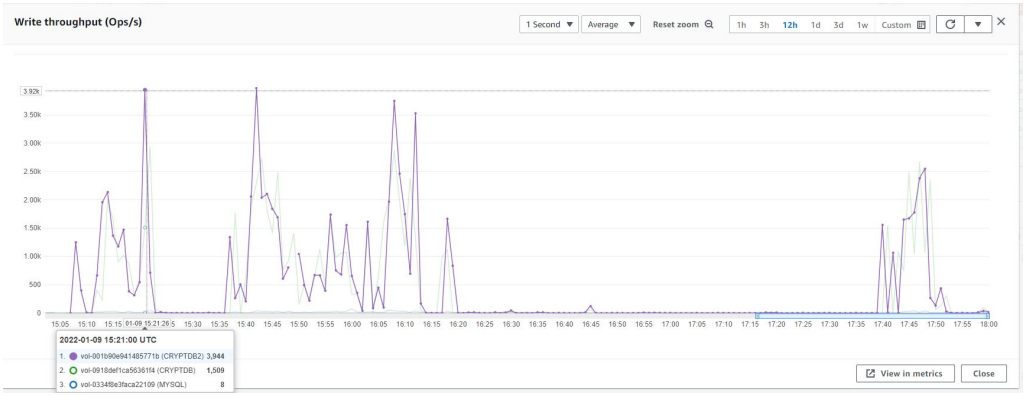
En las Figuras 4.6a 4.6b denotan que las operaciones de escritura por segundo de la unidad de almacenamiento son parecidas en las Instancias de CryptDB y Mysql.

4.3.5.4. Operaciones de Lectura por segundo

En las Figuras 4.7a 4.7b denotan que la mayoría de operaciones de lectura por segundo de la unidad de almacenamiento están en la instancia de Mysql. En la instancia de CryptDB las operaciones de lectura son mínimas por lo cual se descarta un problema en el disco.



(a) Operaciones de Escritura por segundo sin CryptDB con 10,50 y 100 usuarios



(b) Operaciones de Escritura por segundo con CryptDB con 10, 50 y 100 usuarios

Figura 4.6: Operaciones de Escritura por segundo

4.3.5.5. Bytes de Escritura (Bytes)

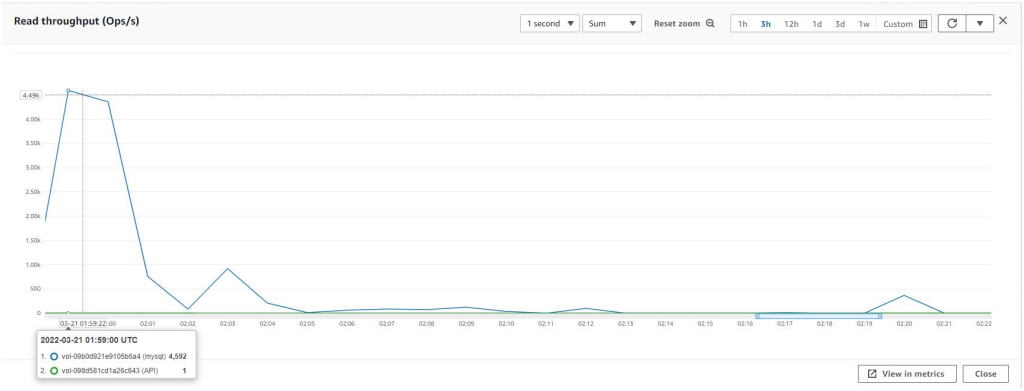
En las Figuras 4.8a 4.8b denotan que el ancho de banda de escritura de la unidad de almacenamiento son parecidas en las Instancias de CryptDB y Mysql.

4.3.5.6. Bytes de Lectura (Bytes):

En las Figuras 4.9a 4.9b denotan que la mayoría ancho de banda de lectura de la unidad de almacenamiento están en la instancia de Mysql. En la instancia de CryptDB las operaciones de lectura son mínimas por lo cual se descarta un problema en el disco.

4.3.5.7. Operaciones promedio Encolada

En la Figura 4.10a se evidencia que las operaciones encoladas son casi inexistentes para el nodo que tiene CryptDB. La línea azul en la Figura 4.10a



(a) Operaciones de Lectura por segundo sin CryptDB con 10,50 y 100 usuarios

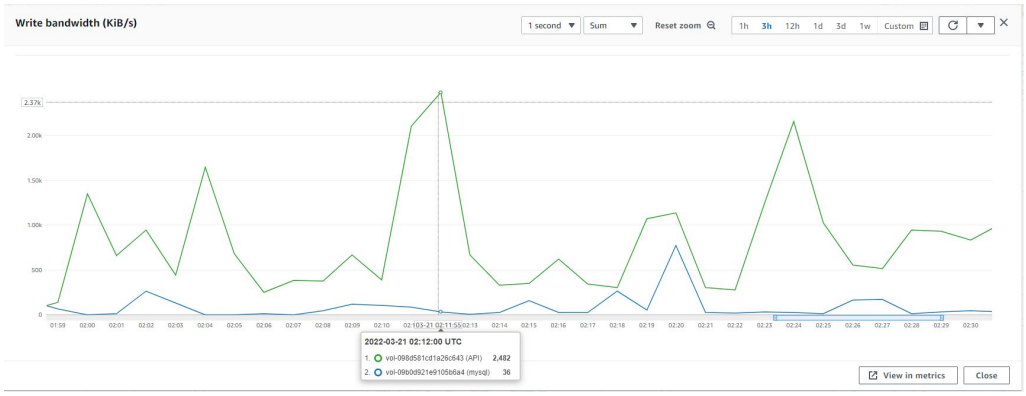
(b) Operaciones de Lectura por segundo con CryptDB con 10,50 y 100 usuarios

Figura 4.7: Operaciones de Lectura por segundo

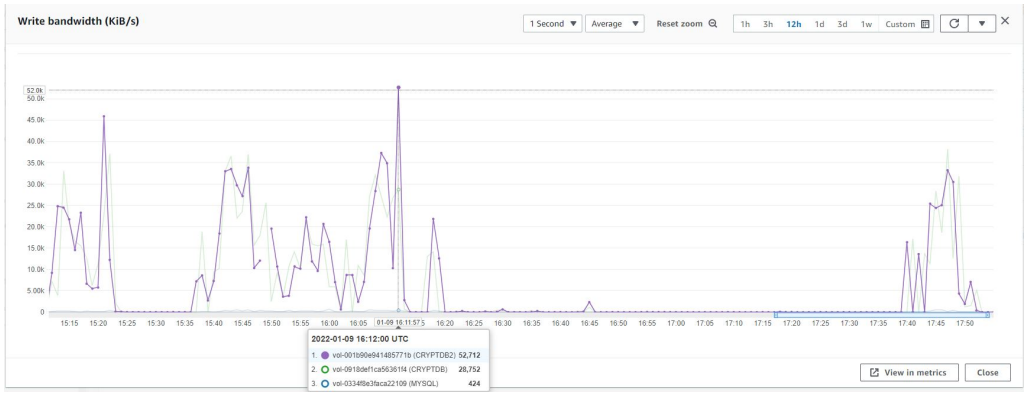
pertenece al nodo que tiene instalado MYSQL.

4.3.5.8. Conclusión de Pruebas de Rendimiento

Las pruebas de rendimiento han demostrado que la arquitectura ACCE sin duda ayuda a reducir los tiempos de respuestas en al menos un 50 % en todos los casos de usuarios concurrentes(10,50,100). El tener un servidor extra en AWS el cuál tiene instalado CryptDB es una gran ayuda para reducir el tiempo de respuesta en cuanto a lectura. Lo que llama la atención es que la instancia de CryptDB nunca alcanza ni el 20 % de consumo del CPU, lo que nos lleva a pensar que CryptDB no está optimizado para alta concurrencia.



(a) Bytes de Escritura sin CryptDB con 10, 50 y 100 usuarios



(b) Bytes de Escritura con CryptDB con 10, 50 y 100 usuarios

Figura 4.8: Bytes de Escritura (Bytes)

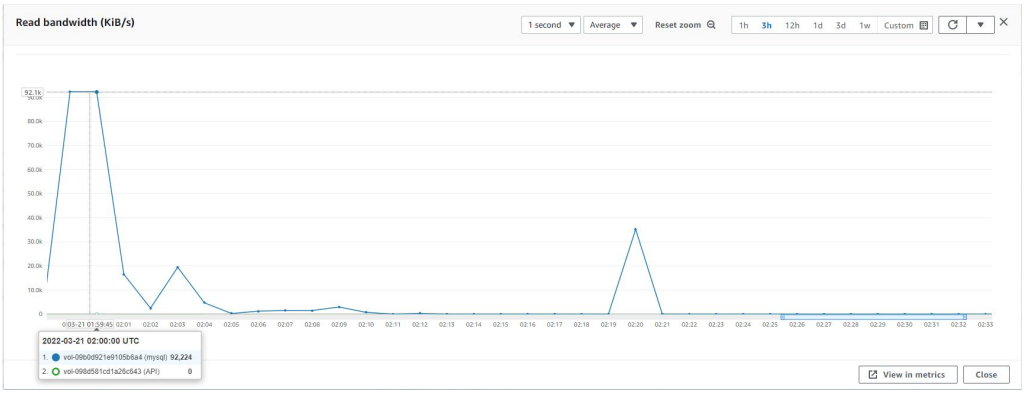
4.4. Discusión

4.4.1. Ventajas de ACCE en Reducción de Latencia

Los resultados confirman la eficacia de la arquitectura ACCE en la reducción significativa de la latencia, especialmente en entornos que utilizan CryptDB. La inclusión de un balanceador de carga y una segunda instancia de CryptDB ha resultado en una mejora sustancial, reduciendo los tiempos de respuesta en un 50 %.

4.4.2. Limitaciones en la Optimización para Alta Concurrencia

Es notable que CryptDB no consume más del 20 % del CPU, lo cual no sólo indica que la herramienta no está optimizada para la alta concurrencia, sino que



(a) Bytes de Escritura sin CryptDB con 10, 50 y 100 usuarios

(b) Bytes de Escritura con CryptDB con 10, 50 y 100 usuarios

Figura 4.9: Bytes de Lectura (Bytes)

también sugiere que hay margen para mejorar la eficiencia de la arquitectura si se agregaran más nodos. No obstante, la agregación de más nodos podría no ser la solución más rentable ni eficiente.

4.4.3. Alternativas a CryptDB

Dada la falta de optimización para alta concurrencia en CryptDB, sería prudente considerar otras herramientas más adaptadas a estas necesidades. Opciones como encriptación de datos en reposo (Data at Rest Encryption) podrían ofrecer un equilibrio entre seguridad y rendimiento.

4.4.4. Documentación y Mantenimiento

La calidad de la documentación es un factor crítico para la adopción de cualquier tecnología en un entorno empresarial. La documentación pobre de CryptDB

(a) Operaciones encoladas con CryptDB con 10, 50 y 100 usuarios

Figura 4.10: Bytes de Lectura (Bytes)

plantea desafíos en su implementación y mantenimiento, lo que podría incurrir en costos adicionales para la empresa.

4.4.5. Necesidad de Expertise en AWS

El uso efectivo de AWS para el balanceo de carga y la creación de instancias adicionales destaca la necesidad de tener un conocimiento sólido en servicios de nube para implementar una arquitectura eficiente. Esta es una consideración importante para las organizaciones que buscan adoptar una solución basada en la nube.

4.4.6. Conclusión de la Discusión

Los resultados de las pruebas y el análisis subsiguiente hacen una contribución valiosa al cuerpo de conocimiento sobre la viabilidad de CryptDB en entornos empresariales. Aunque la herramienta tiene sus méritos en la protección de la privacidad de los datos, sus limitaciones en rendimiento, documentación y requisitos de mantenimiento deben ser cuidadosamente considerados. Las organizaciones deberían evaluar la idoneidad de CryptDB en el contexto de sus requisitos específicos y estar preparadas para invertir en recursos adicionales para su implementación y mantenimiento eficaz.

4.5. Modelo de Amenazas

CryptDB es un sistema de gestión de bases de datos cifradas que proporciona un alto nivel de seguridad para los datos almacenados. La arquitectura propuesta busca mejorar la eficiencia en el rendimiento de CryptDB, pero es crucial entender los posibles riesgos y amenazas que podrían comprometer la integridad y confidencialidad de la información. Este modelo de amenazas tiene como objetivo identificar, clasificar y proponer estrategias de mitigación contra estas amenazas potenciales.

4.5.1. Tipos de Amenazas

- **Suplantación de identidad (Spoofing)**: Los atacantes pueden intentar falsificar su identidad para ganar acceso no autorizado a los datos [49] [50].
- **Manipulación (Tampering)**: Existe el riesgo de que un adversario inyecte datos maliciosos o manipule el código para afectar el funcionamiento de CryptDB [50].
- **Repudio (Repudiation)**: En ausencia de auditoría adecuada, un atacante podría negar las acciones realizadas en la base de datos [50].
- **Divulgación de Información (Information Disclosure)**: Las técnicas de cifrado podrían verse comprometidas, resultando en una posible fuga de información [50].
- **Denegación de Servicio (DoS)**: Ataques que buscan hacer que el sistema sea inaccesible para los usuarios legítimos [50].

4.5.2. Estrategias de Mitigación

Las siguientes estrategias de mitigación se proponen para abordar las amenazas identificadas:

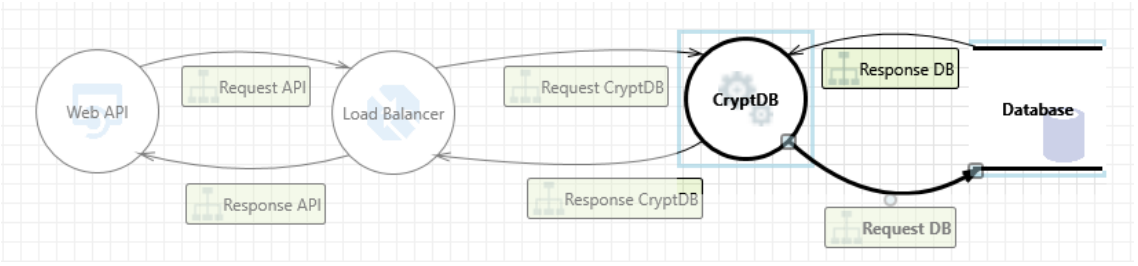


Figura 4.11: Modelo de amenazas global

- **Autenticación Fuerte**: Implementar mecanismos de autenticación de múltiples factores para dificultar la suplantación de identidad [50].

- **Validación de Entradas**: Utilizar técnicas de validación de datos de entrada para prevenir ataques de inyección SQL[51].

- **Auditoría y Logging**: Mantener registros detallados de todas las operaciones para prevenir el repudio [50].

- **Encriptación Robusta**: Utilizar algoritmos de cifrado seguros y probados para mitigar riesgos de divulgación de información [50].

- **Resistencia a DoS**: Implementar técnicas de limitación de tasa y balanceo de carga para mitigar los ataques de Denegación de Servicio [50].

4.5.3. Amenazas

Utilizamos el *Threat Modeling Tool* de Microsoft, una herramienta diseñada para identificar y abordar potenciales amenazas en sistemas antes de que sean implementados. Su función principal es modelar y analizar el entorno de un sistema para descubrir vulnerabilidades y proponer soluciones. Su uso es intuitivo: se crea un diagrama del sistema o aplicación en cuestión, y la herramienta evalúa automáticamente los puntos de riesgo, ofreciendo recomendaciones basadas en patrones conocidos de amenazas y soluciones.

En la Figura 4.11 se puede ver como luce la arquitectura propuesta ACCE y las interacciones ya descritas en la herramienta *Threat Modeling Tool* de Microsoft .

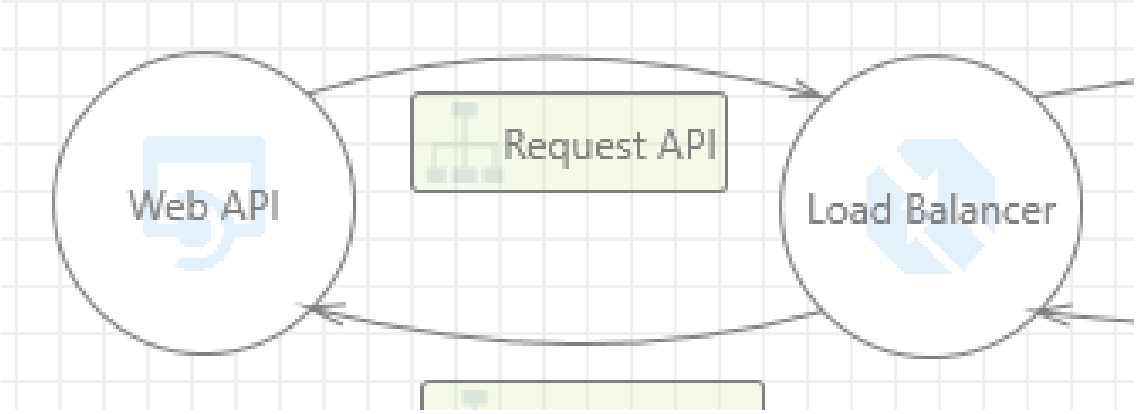


Figura 4.12: Modelo de amenazas Request al API

4.5.3.1. Interacción: Request al API

En la Figura 4.12 se puede ver las interacciones del Request al API

Amenaza 1: Un adversario podría suplantar el servicio o los puntos finales del servicio aprovechando registros DNS CNAME obsoletos y ejecutando un ataque de secuestro de subdominio
Estado: No iniciado
Prioridad: Alta
Categoría: Suplantación de identidad
Descripción: Un adversario podría suplantar el servicio o los puntos finales del servicio aprovechando registros DNS CNAME obsoletos y ejecutando un ataque de secuestro de subdominio.
Posibles mitigaciones: Abordar los registros DNS CNAME obsoletos que mapean nombres de dominio personalizados al nombre de dominio de la instancia del Azure Traffic Manager. En algunos casos, la eliminación de los registros CNAME obsoletos puede ser suficiente, mientras que en otros casos, se debe mantener el nombre de dominio de la instancia del Azure Traffic Manager para prevenir ataques de secuestro de subdominio. Referencia: https://aka.ms/tmt-th178
Fase SDL: Implementación

.5

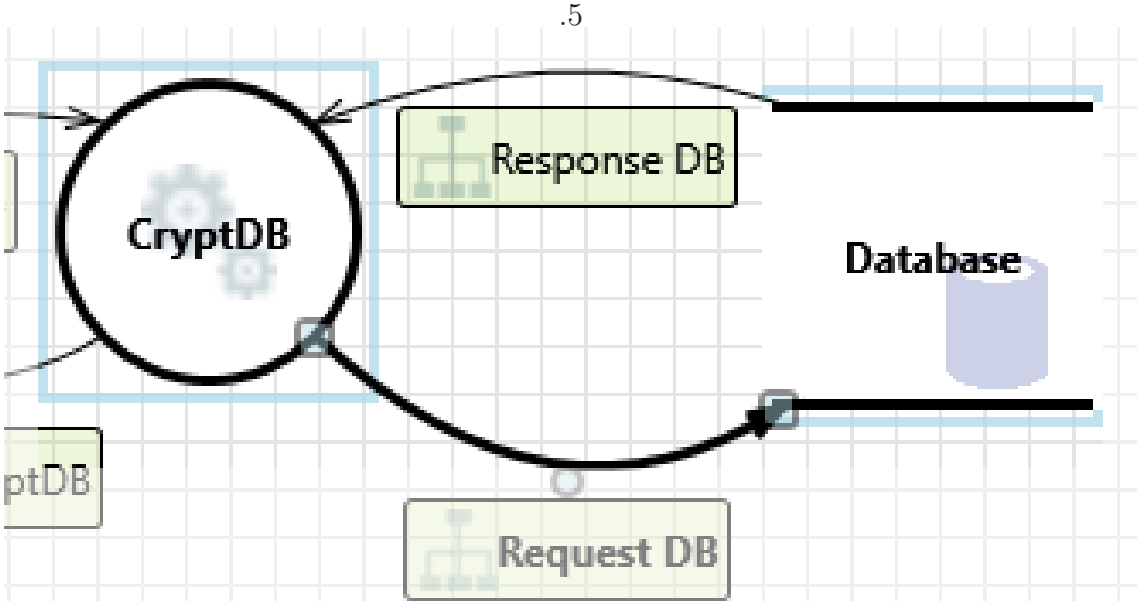


Figura 4.13: Modelo de amenazas Request a la DB

4.5.3.2. Interacción: Request a la DB

En la Figura 4.13 se puede ver las interacciones del Request a la DB **Amenaza 2: Un adversario podría aprovechar la falta de sistemas de monitoreo y desencadenar tráfico anómalo hacia la base de datos**

Estado: No iniciado

Prioridad: Alta

Categoría: Manipulación

Descripción: Un adversario podría aprovechar la falta de detección y prevención de actividades anómalas en la base de datos y desencadenar tráfico anómalo hacia la base de datos.

Posibles mitigaciones: Habilitar la detección de amenazas en la base de datos SQL de Azure. Referencia: https://aka.ms/tmtauditlog#threat-detection

Fase SDL: Implementación

Amenaza 3: Un adversario puede alterar elementos críticos de seguridad en la base de datos y negar la acción

Estado: No iniciado

Prioridad: Alta

Categoría: Manipulación

Descripción: Un adversario puede alterar elementos críticos de seguridad en la base de datos y negar la acción.

Posibles mitigaciones: Un adversario puede alterar elementos críticos de seguridad en la base de datos y negar la acción.

Fase SDL: Implementación

Amenaza 4: Un adversario puede negar acciones en la base de datos debido a la falta de auditoría

Estado: No iniciado

Prioridad: Media

Categoría: Repudio

Descripción: El registro adecuado de todos los eventos de seguridad y las acciones del usuario establece rastreabilidad en un sistema y niega cualquier posible problema de repudio. En ausencia de controles de auditoría y registro adecuados, sería imposible implementar cualquier responsabilidad en un sistema.

Posibles mitigaciones: Asegurarse de que la auditoría de inicio de sesión esté habilitada en SQL Server. Referencia: https://aka.ms/tmtauditlog#identify-sensitive-entities

Fase SDL: Implementación

Amenaza 5: Un adversario puede obtener acceso a datos sensibles mediante la realización de una inyección SQL

Estado: No iniciado

Prioridad: Alta

Categoría: Divulgación de Información

Descripción: La inyección SQL es un ataque en el que se inserta código malicioso en cadenas que posteriormente se pasan a una instancia de SQL Server para su análisis y ejecución. La forma principal de inyección SQL consiste en la inserción directa de código en variables de entrada del usuario que se concatenan con comandos SQL y se ejecutan. Un ataque menos directo inyecta

código malicioso en cadenas destinadas a ser almacenadas en una tabla o como metadatos. Cuando las cadenas almacenadas se concatenan posteriormente en un comando SQL dinámico, se ejecuta el código malicioso.

Posibles Mitigaciones: Asegurarse de que la auditoría de inicio de sesión esté habilitada en SQL Server. Referencia: https://aka.ms/tmtauditlog#identify-sensitive-entities Asegurarse de que se utilicen cuentas con los mínimos privilegios para conectarse al servidor de base de datos. Referencia: https://aka.ms/tmtauthz#privileged-server Habilitar la detección de amenazas en la base de datos SQL de Azure. Referencia: https://aka.ms/tmtauditlog#threat-detection No utilizar consultas dinámicas en procedimientos almacenados. Referencia: https://aka.ms/tmtinputval#stored-proc

Fase SDL: Implementación

Amenaza 6: Un adversario puede obtener acceso a datos personales sensibles (PII) o información de alto impacto comercial (HBI) en la base de datos

Estado: No iniciado

Prioridad: Baja

Categoría: Divulgación de Información

Descripción: Controles adicionales como el Cifrado de Datos Transparente, Cifrado a Nivel de Columna, EKM, etc., proporcionan un mecanismo de protección adicional para datos PII o HBI de alto valor.

Posibles Mitigaciones: Mitigación ya implementada por diseño mediante el uso de CryptDB

Fase SDL: Implementación

Amenaza 7: Un adversario puede obtener acceso no autorizado a la base de datos debido a reglas de autorización laxas

Estado: No iniciado

Prioridad: Alta

Categoría: Elevación de Privilegios

Descripción: El acceso a la base de datos debe estar configurado con roles y

privilegios basados en el principio de mínimo privilegio y necesidad de saber.
Posibles Mitigaciones: Asegurar que se utilicen cuentas de mínimo privilegio para conectarse al servidor de la base de datos. Referencia: https://aka.ms/tmtauthz#privileged-server Implementar Seguridad a Nivel de Fila (RLS) para evitar que los inquilinos accedan a los datos de los demás. Referencia: https://aka.ms/tmtauthz#rls-tenants El rol de administrador del sistema (Sysadmin) solo debe tener usuarios válidos y necesarios. Referencia: https://aka.ms/tmtauthz#sysadmin-users
Fase SDL: Implementación

Amenaza 8: Un adversario puede obtener acceso no autorizado a la base de datos debido a la falta de protección de acceso de red
Estado: No iniciado
Prioridad: Alta
Categoría: Elevación de Privilegios
Descripción: Si no hay restricciones a nivel de red o del cortafuegos del host para acceder a la base de datos, cualquier persona puede intentar conectarse a la base de datos desde una ubicación no autorizada.
Posibles Mitigaciones: Configurar un cortafuegos de Windows para el acceso al motor de la base de datos. Referencia: https://aka.ms/tmtconfigmgmt#firewall-db
Fase SDL: Implementación

4.5.3.3. Interacción: Respuesta del API

En la Figura 4.14 se puede ver las interacciones de la Respuesta del API
Amenaza 9: Un adversario podría obtener acceso no autorizado a la Web API debido a controles de acceso deficientes
Estado: No iniciado
Prioridad: Alta
Categoría: Elevación de Privilegios
Descripción: Un adversario podría obtener acceso no autorizado a la Web API

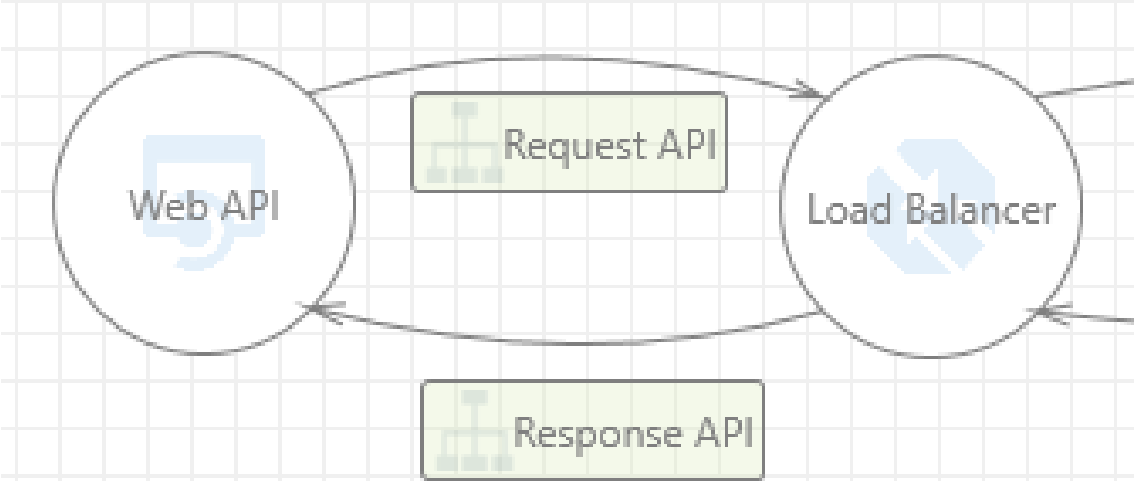


Figura 4.14: Modelo de amenazas Respuesta del API

debido a controles de acceso deficientes.

Posibles Mitigaciones: Implementar un mecanismo de autorización adecuado en ASP.NET Web API. Referencia: https://aka.ms/tmtauthz#authz-aspnet

Fase SDL: Implementación

Amenaza 10: Un adversario podría obtener acceso a información sensible a través de mensajes de error en una API

Estado: No iniciado

Prioridad: Alta

Categoría: Divulgación de Información

Descripción: Un adversario podría obtener acceso a datos sensibles, como los siguientes, a través de mensajes de error detallados: nombres de servidores, cadenas de conexión, nombres de usuario, contraseñas, procedimientos SQL, detalles de fallos de SQL dinámico, seguimiento de pila y líneas de código, variables almacenadas en memoria, ubicaciones de unidades y carpetas, puntos de instalación de aplicaciones, configuraciones del host, y otros detalles internos de la aplicación.

Posibles Mitigaciones: Asegurar que se realice un manejo adecuado de excepciones en ASP.NET Web API. Referencia: https://aka.ms/tmtxmgmt#exception

Fase SDL: Implementación

Amenaza 11: Un adversario podría obtener acceso a datos sensibles interceptando el tráfico hacia la Web API

Estado: No iniciado

Prioridad: Alta

Categoría: Divulgación de Información

Descripción: Un adversario podría obtener acceso a datos sensibles interceptando el tráfico hacia la Web API.

Posibles Mitigaciones: Forzar todo el tráfico a las Web API a través de una conexión HTTPS. Referencia: https://aka.ms/tmtcommsec#webapi-https

Fase SDL: Implementación

Amenaza 12: Un adversario podría obtener acceso a datos sensibles almacenados en los archivos de configuración de la Web API

Estado: No iniciado

Prioridad: Media

Categoría: Divulgación de Información

Descripción: Un adversario podría obtener acceso a los archivos de configuración. Y si en ellos se almacenan datos sensibles, estos estarían comprometidos.

Posibles Mitigaciones: Encriptar secciones de los archivos de configuración de la Web API que contengan datos sensibles. Referencia: https://aka.ms/tmtconfigmgmt#config-sensitive

Fase SDL: Implementación

Amenaza 13: Un atacante podría negar un acto malicioso en una API, lo que lleva a problemas de repudio

Estado: No iniciado

Prioridad: Alta

Categoría: Repudio

Descripción: Un atacante podría negar un acto malicioso en una API, lo que lleva a problemas de repudio.

Posibles Mitigaciones: Asegurar que la auditoría y el registro estén aplicados

en la Web API. Referencia: https://aka.ms/tmtauditlog#logging-web-api
Fase SDL: Diseño

Amenaza 14: Un adversario podría suplantar el equilibrador de carga y obtener acceso a la Web API
Estado: No iniciado
Prioridad: Alta
Categoría: Suplantación de identidad
Descripción: Si no se implementa una autenticación adecuada, un adversario podría suplantar un proceso de origen o entidad externa y obtener acceso no autorizado a la aplicación web.
Posibles Mitigaciones: Asegurarse de que se utilicen técnicas de autenticación estándar para proteger las Web APIs. Referencia: https://aka.ms/tmtauthn#authn-secure-api
Fase SDL: Diseño

Amenaza 15: Un adversario podría inyectar entradas maliciosas en una API y afectar los procesos posteriores
Estado: No iniciado
Prioridad: Alta
Categoría: Manipulación
Descripción: Un adversario podría inyectar entradas maliciosas en una API y afectar los procesos posteriores.
Posibles Mitigaciones: Asegurarse de que se realice la validación del modelo en los métodos de la Web API. Referencia: https://aka.ms/tmtinputval#validation-api. Implementar la validación de entradas en todos los parámetros de tipo cadena aceptados por los métodos de la Web API. Referencia: https://aka.ms/tmtinputval#string-api
Fase SDL: Implementación

Amenaza 16: Un adversario puede obtener acceso a datos sensibles me-

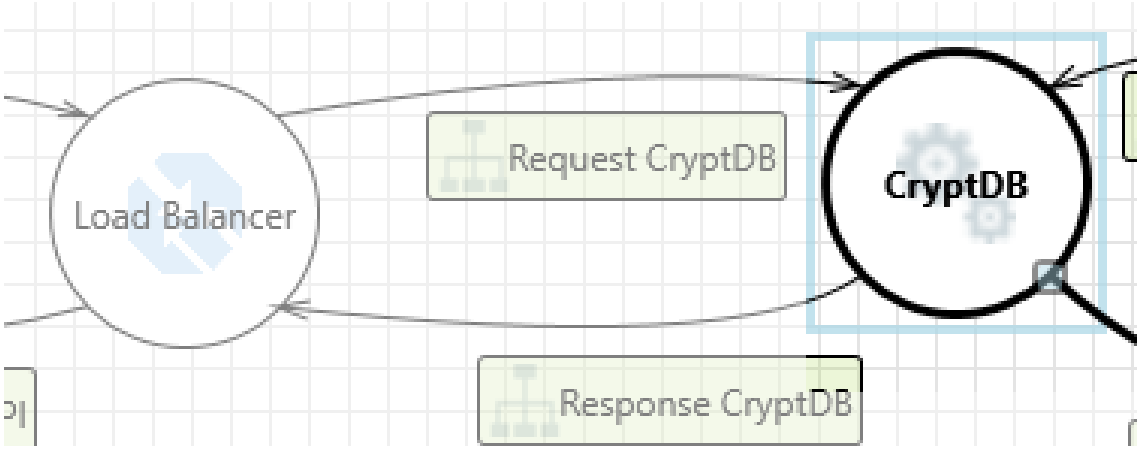


Figura 4.15: Modelo de amenazas Respuesta de CryptDB

diante la realización de una inyección SQL a través de la Web API

Estado: No iniciado

Prioridad: Alta

Categoría: Manipulación

Descripción: La inyección SQL es un ataque en el cual se inserta código malicioso en cadenas que posteriormente son pasadas a una instancia de SQL Server para su análisis y ejecución. La forma principal de inyección SQL consiste en la inserción directa de código en variables de entrada del usuario que se concatenan con comandos SQL y se ejecutan. Un ataque menos directo inyecta código malicioso en cadenas que están destinadas para ser almacenadas en una tabla o como metadatos. Cuando las cadenas almacenadas son posteriormente concatenadas en un comando SQL dinámico, el código malicioso se ejecuta.

Posibles Mitigaciones: Asegurarse de que se utilicen parámetros seguros en la Web API para el acceso a datos. Referencia: https://aka.ms/tmtinputval#typesafe-api

Fase SDL: Implementación

4.5.3.4. Interacción: Respuesta de CryptDB

En la Figura 4.15 se puede ver las interacciones de la Respuesta de CryptDB

Amenaza 17: Un adversario podría suplantar el servicio o los puntos finales

del servicio aprovechando registros DNS CNAME obsoletos y ejecutando un ataque de secuestro de subdominio
Estado: No iniciado
Prioridad: Alta
Categoría: Suplantación de identidad
Descripción: Un adversario podría suplantar el servicio o los puntos finales del servicio aprovechando registros DNS CNAME obsoletos y ejecutando un ataque de secuestro de subdominio.
Posibles Mitigaciones: Abordar los registros DNS CNAME obsoletos que mapean nombres de dominio personalizados al nombre de dominio de la instancia del Azure Traffic Manager. En algunos casos, la eliminación de los registros CNAME obsoletos puede ser suficiente, mientras que en otros casos, se debe mantener el nombre de dominio de la instancia del Azure Traffic Manager para prevenir ataques de secuestro de subdominio. Referencia: https://aka.ms/tmt-th178
Fase SDL: Implementación

4.5.4. Resumen del Modelo de Amenazas

El modelado de amenazas, tal como se presentó en el contexto de CryptDB y la arquitectura ACCE propuesta, es un enfoque esencial para anticipar y abordar los potenciales riesgos que podrían comprometer la integridad y confidencialidad de la información. Mediante este proceso, se identificaron diferentes tipos de amenazas, tales como suplantación de identidad, manipulación de datos, repudio de acciones, divulgación de información y denegación de servicio. Cada una de estas amenazas tiene un impacto distinto en la arquitectura de CryptDB y, por tanto, requiere estrategias específicas de mitigación.
Dentro del marco propuesto, las estrategias de mitigación desempeñan un papel crucial para contrarrestar las amenazas identificadas. Desde la implementación de autenticación fuerte hasta la resistencia a ataques de Denegación de Servicio, el objetivo es garantizar que CryptDB no solo mantenga su promesa de seguridad, sino que también funcione con un alto rendimiento. Es fundamental aplicar este

modelo de amenazas a lo largo de las fases de implementación de proyectos de software para garantizar una implementación robusta y segura de CryptDB con la arquitectura propuesta.

Capítulo 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Los resultados de las pruebas realizadas demuestran que la arquitectura propuesta de “Ambiente Cifrado Con Escalamiento(ACCE)” puede ayudar a reducir significativamente la latencia en aplicaciones que utilizan CryptDB. La implementación de un balanceador de carga y una instancia adicional de CryptDB permitió distribuir la carga entre dos nodos, lo que resultó en una reducción del 50 % en la latencia. Sin embargo, se observó que CryptDB no utilizó ni el 20 % del CPU, lo que sugiere que CryptDB no está optimizado para la alta concurrencia. Esto sugiere que si se añaden más nodos a la arquitectura, el tiempo de latencia podría bajar un poco más.

En lugar de seguir añadiendo nodos a la arquitectura, la mejor opción sería considerar cambiar CryptDB por otra herramienta que sea más adecuada para proyectos con alta concurrencia, o utilizar una herramienta que ofrezca encriptación de datos en reposo (Data at Rest Encryption) para garantizar la privacidad y seguridad de los datos.

Otro hallazgo importante fue que la documentación de CryptDB fue evaluada como pobre, lo que podría ser un obstáculo para su adopción en empresas. El costo de mantenimiento de CryptDB también podría ser elevado debido a la falta de documentación y a que la herramienta no está diseñada para una alta concurrencia.

A pesar de estos hallazgos, los resultados también sugieren que CryptDB es una herramienta útil para garantizar la privacidad de los datos. Es importante considerar cuidadosamente la implementación de CryptDB en función de las necesidades específicas de una empresa y su capacidad de inversión en la herramienta.

Además de lo mencionado anteriormente, es importante destacar que el

conocimiento en AWS (Amazon Web Services) es esencial para implementar una arquitectura escalable y eficiente en la nube. En este caso, se utilizó AWS para crear una instancia EC2 para CryptDB y otra para MySQL, además de un balanceador de carga para distribuir la carga entre ambas instancias.

En resumen, aunque CryptDB presenta algunas limitaciones, es una herramienta que en su tiempo fue pionera y es un hito para garantizar la privacidad de los datos. Sin embargo, se debe tener en cuenta que la implementación de CryptDB en un entorno empresarial puede ser costosa y requerir un esfuerzo significativo de mantenimiento debido a la falta de documentación y a la falta de optimización para la alta concurrencia. Aunque se logró demostrar la eficacia de la arquitectura propuesta, es importante tener en cuenta las limitaciones de las herramientas utilizadas y considerar alternativas más adecuadas en función de las necesidades del proyecto. Asimismo, el conocimiento y experiencia en AWS y otras tecnologías relacionadas son fundamentales para implementar arquitecturas escalables y eficientes en la nube.

5.2. Recomendaciones

Al utilizar CryptDB en un proyecto empresarial, es importante tener en cuenta sus limitaciones y considerar utilizar otras alternativas que estén más optimizadas para la utilización máxima del CPU y que ofrezcan una documentación más completa. En caso de no encontrar una alternativa viable, es importante contratar a expertos en el tema para implementar y mantener CryptDB.

Al considerar CryptDB en un proyecto real, es importante considerar que el motor de base de datos de MySQL necesita ser de versiones antiguas, lo cual puede generar vulnerabilidades de seguridad. En este caso, se recomienda utilizar otras alternativas que sean más seguras y estén actualizadas con los últimos parches de seguridad.

Si se maneja información sensible, como registros médicos, es importante tener en cuenta los problemas de datos corruptos que ha presentado CryptDB en el pasado. En este caso, se recomienda utilizar alternativas que sean más confiables

y que cuenten con herramientas de respaldo y recuperación de datos en caso de alguna falla.

Si se desea implementar la arquitectura propuesta ACCE en un proyecto, es importante considerar que esta tiene limitaciones en cuanto a la escritura de información. Para solucionar este problema, se recomienda implementar patrones de software como CQRS, que permiten diferenciar la escritura de la lectura y así hacer que la arquitectura sea funcional para ambos casos.

CQRS es un patrón de diseño de software que significa Çommand Query Responsibility Segregation"(Segregación de la responsabilidad de comandos y consultas). Es una técnica que separa las operaciones de escritura (comandos) de las operaciones de lectura (consultas) en sistemas de información.

En lugar de tener un único modelo de datos que maneje ambas operaciones, CQRS propone tener dos modelos de datos separados, uno para comandos y otro para consultas. El modelo de comandos se encarga de actualizar y modificar los datos, mientras que el modelo de consultas se encarga de leer los datos.

De esta manera, el modelo de comandos puede estar optimizado para la escritura, mientras que el modelo de consultas puede estar optimizado para la lectura. Esto permite que cada modelo pueda ser escalado y optimizado de forma independiente, lo cual puede mejorar el rendimiento y la eficiencia del sistema.

En el contexto de la arquitectura ACCE propuesta, implementar el patrón CQRS puede permitir que la arquitectura sea funcional para ambas operaciones, tanto la lectura como la escritura de información. Esto podría ayudar a mejorar la eficiencia y la escalabilidad del sistema en general.

Para implementar una arquitectura escalable y eficiente como la propuesta ACCE, es importante contar con un buen conocimiento en AWS (Amazon Web Services) y sus servicios. AWS ofrece una amplia variedad de herramientas y servicios que pueden ser utilizados para mejorar la eficiencia y seguridad de la arquitectura. Se recomienda capacitarse en AWS y adquirir experiencia en la implementación de arquitecturas en la nube antes de implementar la arquitectura ACCE en un proyecto empresarial. Además, es importante tener en cuenta los costos asociados con el uso de los servicios de AWS y optimizar su uso para reducir los

gastos operativos del proyecto.

Anexos

Anexo A

Modelo Relacional MIMIC III

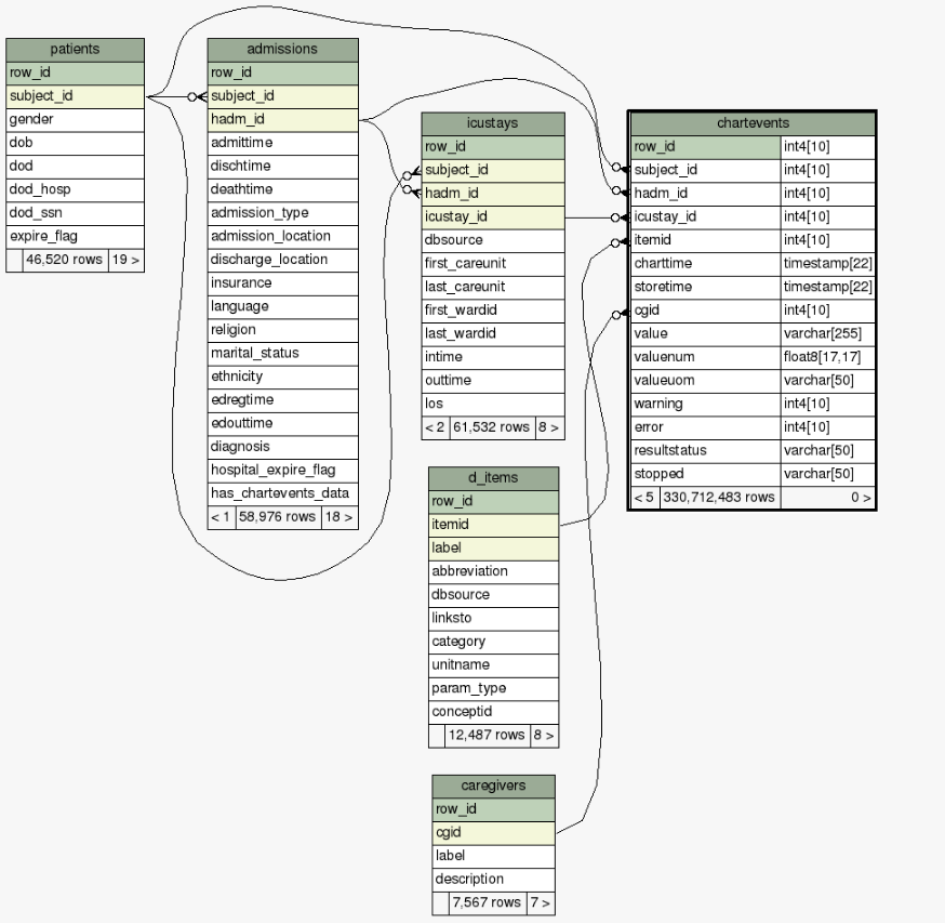


Figura A.1: Diagrama de relaciones para tabla ChartEvents

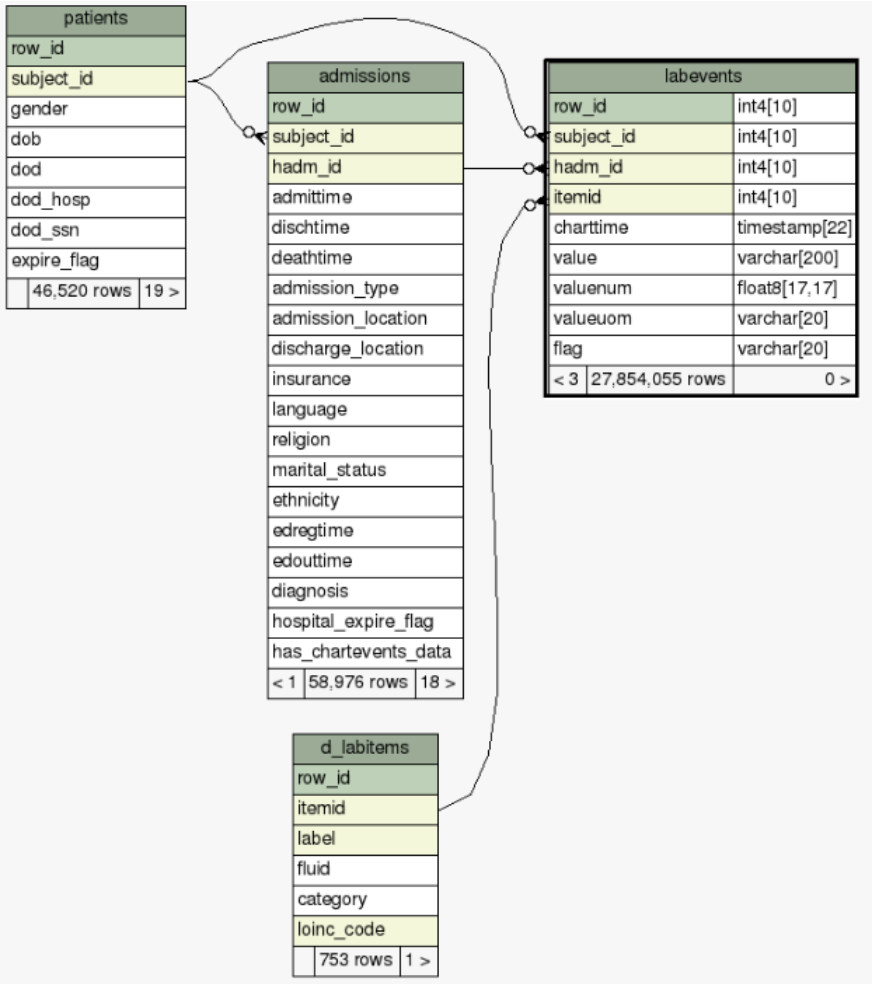


Figura A.2: Diagrama de relaciones para tabla LabEvents

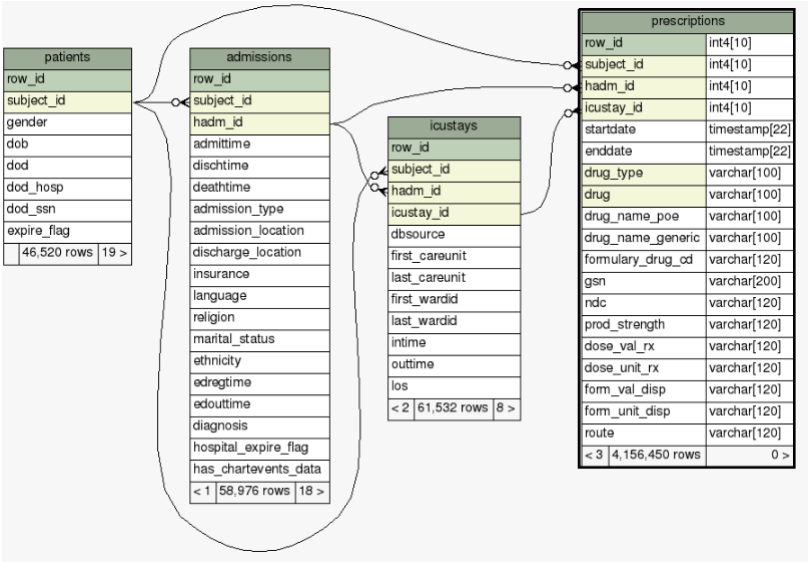


Figura A.3: Diagrama de relaciones para tabla Prescriptions

Anexo B

Encuestas, Código fuente, recursos y diagramas

B.1. Encuesta Doctor UCI

B.1.1. Preguntas

- ¿Qué datos del paciente son importantes al momento de revisar un paciente en UCI?
- ¿Con qué frecuencia al día se revisan esos datos?
- ¿La revisión de la medicación está incluida en los datos del paciente?
- ¿Con qué frecuencia se revisa la medicación?

B.1.2. Respuestas

- Los datos que se revisan de un paciente son: frecuencia cardiaca, presión arterial, frecuencia frecuencia respiratoria, glucosa, hematocritos, CO2 arterial, entre otros dependiendo el caso del paciente
- Esto también dependerá del caso del paciente, pero por lo general es mínimo una o dos veces al día.
- Si
- Igual se lo revisa al mismo momento que al revisar los datos del paciente.

B.2. Preguntas y respuestas de parámetros de calificación de calidad de documentación de software

Aquí están las preguntas y respuestas, posibles respuestas “SI”, “NO” y el porque

1. **Precisión:**

    - ¿Proporciona la documentación información precisa y actualizada sobre el software? (No) - La documentación no proporciona información generalmente precisa y actualizada.
    - ¿Pudo encontrar todos los términos y conceptos técnicos explicados en la documentación? (No) - Los términos y conceptos técnicos se explican adecuadamente, aunque en algunos casos puede ser necesario hacer una búsqueda para encontrarlos.

2. **Claridad:**

    - ¿Pudo correr el programa siguiendo la documentación? (Si) - Las instrucciones y explicaciones básicas estaban claras de como correr el programa por primera vez.
    - ¿Encontró las instrucciones y explicaciones en la documentación fáciles de entender? (No) - Las instrucciones y explicaciones de como conectas CryptDB y MySQL no se presentan de manera clara y pueden ser un poco confusas. Se tuvo que leer el código fuente para poder entenderlo
    - ¿Fue el lenguaje utilizado en la documentación claro y conciso? (No) - El lenguaje utilizado en la documentación no es tan claro, a veces puede ser un poco confuso o poco claro.

3. **Consistencia:**

    - ¿Fueron consistentes los términos y la presentación en toda la documentación? (Si) - Los términos y la presentación son consistentes en la mayoría de la documentación, pese a que es corta. Aunque en algunos casos puede haber algunas inconsistencias menores.
    - ¿Se explicaron los mismos conceptos de la misma manera en toda la documentación? (No) - A veces los mismos conceptos se explican de

diferentes maneras en diferentes partes de la documentación, lo que puede ser un poco confuso.

4. **Legibilidad:**

    - ¿Fue fácil leer el texto en general? (No) - El texto no es fácil de leer en general, es un un poco denso o técnico y mucho de este está en desorden.

5. **Estructuración:**

    - ¿Está la documentación estructurada de manera coherente y fácil de navegar? (No) - La estructuración de la documentación no coherente ni fácil de navegar, ya que los archivos están en una carpeta con su contenido en desorden.

6. **Comprensibilidad:**

    - ¿Pudo entender fácilmente la función y el propósito del software a partir de la documentación? (No) - Si se entiende el propósito, pero entender la función de cada componente es un poco más difícil y muchas veces es necesario leer el código fuente para entender que hace
    - ¿Pudo encontrar rápidamente las respuestas a sus preguntas en la documentación? (No) - No, realmente es difícil encontrar algunas respuestas. Es necesario leer el código fuente para entender lo que hace

Anexo C

Glosario

Cifrado: Proceso de convertir mensajes o datos en un formato no legible por nadie más, excepto por la persona para quien fue destinado el mensaje.

Encriptación: Véase Cifrado.

Desencriptación: Proceso de convertir datos encriptados en su formato original legible.

Algoritmos simétricos: Algoritmos de encriptación que utilizan la misma llave para encriptar y desencriptar el contenido.

Algoritmos asimétricos: Algoritmos de encriptación que utilizan una llave pública para encriptar el mensaje y una llave privada para desencriptar el mensaje.

Cifrado Determinista: Tipo de cifrado que produce el mismo texto cifrado para un texto plano dado, siempre y cuando se utilice la misma llave.

Cifrado Probabilístico: Tipo de cifrado que produce diferentes textos cifrados para un texto plano dado, incluso cuando se utiliza la misma llave, introduciendo aleatoriedad en el proceso.

Cifrado Homomórfico: Tipo de encriptación que permite realizar cálculos sobre datos cifrados, produciendo un resultado que, cuando se descifra, coincide con el resultado que se habría obtenido si los cálculos se hubieran realizado sobre datos sin cifrar.

RSA (Rivest–Shamir–Adleman): Un algoritmo de cifrado de llave pública que tiene propiedades homomórficas.

Homomorfismo: En álgebra, una propiedad que refleja la preservación de la estructura entre dos estructuras algebraicas.

AES (Advanced Encryption Standard): Un estándar de encriptación utilizado ampliamente, conocido también como Rijndael.

CryptDB: Un sistema que proporciona un nivel de seguridad para bases de datos SQL, permitiendo operaciones SQL sobre datos encriptados.

Amazon Web Services (AWS): Una plataforma de servicios de nube que ofrece potencia de cómputo, almacenamiento de bases de datos, entrega de contenido y otras funcionalidades.

Cifrado Dinámico Ajustable: Característica de CryptDB que le permite ajustar el esquema de cifrado basado en las consultas, equilibrando seguridad y eficiencia.

Vinculación de Claves de Cifrado: Método de asegurar que solo los propietarios legítimos de la información puedan descifrar los datos.

Llave Maestra (MK): Clave principal utilizada en CryptDB para una serie de funciones, incluida la anonimización de tablas y nombres de columnas.

Funciones Definidas por el Usuario (UDF): Funciones personalizadas que se pueden definir según las necesidades del usuario o la aplicación.

Random (RND): Esquema de cifrado que proporciona alto nivel de seguridad al generar textos cifrados únicos para el mismo texto plano mediante un proceso probabilístico.

Deterministic (DET): Esquema de cifrado que produce el mismo texto cifrado para un texto plano dado, siempre y cuando se utilice la misma llave.

Order-preserving (OPE): Esquema de cifrado que mantiene el orden de los datos, permitiendo comparaciones de magnitud en datos cifrados.

Homomorphic encryption (HOM): Tipo de cifrado que permite cálculos en datos cifrados, de modo que los resultados, cuando se descifran, coinciden con los resultados de operaciones en datos sin cifrar.

Join (JOIN y OPE-JOIN): Esquemas de cifrado utilizados en CryptDB para permitir comparaciones de igualdad entre columnas y determinar valores cifrados repetidos entre columnas.

Word Search (SEARCH): Esquema de cifrado que permite la búsqueda de palabras en textos cifrados.

Onion Encryption: Mecanismo utilizado por CryptDB que aplica múltiples capas de cifrado, cada una con un nivel diferente de seguridad y funcionalidad.

Criptosistema de Paillier: Esquema de cifrado homomórfico que permite la adición de dos números cifrados y el resultado, cuando se descifra, es la suma de los números originales.

Fully Homomorphic Encryption (FHE): Tipo de cifrado que permite la ejecución de cálculos arbitrarios en datos cifrados sin requerir acceso al texto plano.

Escalamiento Vertical: Proceso de aumentar la capacidad de un único servidor a través de la adición de recursos más potentes, como una CPU más fuerte o más memoria RAM.

Escalamiento Horizontal: Método que implica añadir más servidores que funcionan juntos como un sistema unificado, en lugar de aumentar la capacidad de un único servidor. Este enfoque ayuda a mejorar la redundancia y la disponibilidad.

Proxy: Entidad que actúa como intermediario entre el usuario final y el servidor, que en el contexto de CryptDB, procesa la información encriptando las consultas y desencriptando los resultados.

Amazon Web Services (AWS): (Ya definido anteriormente) Una plataforma de servicios de nube que ofrece potencia de cómputo, almacenamiento de bases de datos, entrega de contenido y otras funcionalidades.

EC2 (Elastic Compute Cloud): Servicio proporcionado por Amazon Web Services que permite a los usuarios alquilar servidores virtuales en los que ejecutar aplicaciones y servicios.

Grupos de Auto Escalamiento: En AWS, grupos que contienen un conjunto de instancias EC2 tratadas como una unidad lógica con fines de escalado automático y gestión. El servicio de Auto Escalamiento ajusta automáticamente el número de instancias EC2 en respuesta a las condiciones actuales.

RDS (Amazon Relational Database Service): Un servicio gestionado que configura y opera bases de datos relacionales en la nube, ofreciendo opciones de escalabilidad y administración automatizada.

Load Balancer: Componente que distribuye automáticamente las aplicaciones entrantes o el tráfico de red a través de múltiples destinos, como instancias, contenedores y direcciones IP.

Application Load Balancer (ALB): Tipo de Load Balancer diseñado para el tráfico HTTP y HTTPS, que opera en la capa 7 y es capaz de enrutar el tráfico basándose en el contenido de la solicitud.

Network Load Balancer (NLB): Tipo de Load Balancer que opera en la capa 4, manejando conexiones TCP y UDP, y es conocido por su alta capacidad y baja latencia.

Classic Load Balancer (CLB): El más antiguo de los Load Balancers en AWS, que opera tanto en la capa 4 como en la capa 7 y proporciona una solución básica de balanceo de carga.

Target Group: En AWS, un conjunto de recursos de destino, como instancias EC2, que trabajan en conjunto con Load Balancers para dirigir el tráfico de red según las reglas definidas.

MIMIC-III: Medical Information Mart for Intensive Care III, una base de datos extensa que contiene información detallada sobre pacientes ingresados en

unidades de cuidados intensivos. Incluye diversos datos como signos vitales, mediciones de laboratorio, medicamentos, observaciones, entre otros.

Signos vitales: Mediciones básicas que indican el estado de las funciones corporales esenciales de un paciente, comúnmente incluyen la temperatura corporal, la presión arterial, el pulso (frecuencia cardíaca) y la respiración.

Unidades de cuidados intensivos (UCI): Departamentos especializados de un hospital que brindan atención médica intensiva y monitoreo a pacientes con condiciones de salud críticas.

Códigos de procedimiento: Códigos numéricos o alfanuméricos utilizados para identificar procedimientos médicos específicos realizados durante la atención de un paciente.

Códigos de diagnóstico: Códigos numéricos o alfanuméricos utilizados para clasificar enfermedades y una amplia variedad de signos, síntomas, hallazgos anormales, quejas, circunstancias sociales y causas externas de lesiones o enfermedades.

Balance de líquidos: Un resumen o registro de la cantidad total de fluido que un paciente ha tomado y eliminado durante un período específico.

Anonimización: Proceso de eliminar o enmascarar información personal identificable para proteger la privacidad del individuo, mientras se mantiene la integridad de los datos para su análisis o uso secundario.

HIPAA (Ley de Responsabilidad y Portabilidad del Seguro de Salud): Ley de EE. UU. que proporciona protecciones de privacidad y seguridad para salvaguardar la información médica de los individuos.

Metodología de investigación: Proceso sistemático y científico utilizado para aplicar técnicas y métodos con el fin de generar nuevo conocimiento.

Metodología experimental: Enfoque de investigación que se basa en experimentos controlados para validar hipótesis o explorar causas y efectos.

Revisión sistemática: Análisis exhaustivo y estructurado de toda la literatura disponible sobre un tema específico, utilizando una metodología predefinida.

Metodología de Kitchenham: Marco estructurado específico para realizar revisiones sistemáticas en el campo de la ingeniería de software.

PRISMA: Acrónimo de "Preferred Reporting Items for Systematic Reviews and Meta-Analyses". Es un conjunto de directrices para reportar revisiones sistemáticas y metaanálisis, especialmente en el campo de la salud.

Meta-análisis: Método estadístico para combinar los resultados de varios estudios independientes sobre un mismo tema, aumentando la potencia estadística y mejorando la estimación del tamaño del efecto.

K6: Herramienta de pruebas de rendimiento para sistemas y software, utilizada para probar cargas de trabajo y evaluar la capacidad de respuesta y estabilidad.

Teorema del residuo chino: Método matemático para resolver sistemas de congruencias simultáneas. A menudo se utiliza en la teoría de números y para optimizar cálculos en sistemas informáticos.

Ambiente No Cifrado (ANC): Ambiente de prueba que contiene datos médicos sin cifrar.

Ambiente Cifrado Sin Escalamiento (ACSE): Ambiente de prueba que contiene datos cifrados utilizando CryptDB.

Ambiente Cifrado Con Escalamiento (ACCE): Ambiente de prueba implementado para mejorar el rendimiento del ambiente ACSE.

Escalamiento Vertical: Método de añadir más potencia a una máquina existente, como más CPUs, memoria o almacenamiento.

t3a y t3: Tipos de instancias de AWS de uso general.

SSD (Solid State Disk): Tipo de almacenamiento que utiliza chips de memoria no volátil para guardar datos en lugar de discos magnéticos como en los discos duros tradicionales.

Cluster: Grupo de servidores trabajando juntos para mejorar el rendimiento y la disponibilidad.

UCI (Unidad de Cuidados Intensivos): Un departamento especializado en un hospital que proporciona atención intensiva y monitoreo a pacientes con condiciones críticas.

CSV (Valores Separados por Comas): Un formato de archivo simple usado para almacenar datos tabulares, como los que se encuentran en una hoja de cálculo o una base de datos.

MySQL: Un sistema de gestión de bases de datos relacional de código abierto.

HADM_ID: Identificador único para cada ingreso hospitalario en la base de datos MIMIC-III.

SUBJECT_ID: Identificador único para cada paciente en la base de datos MIMIC-III.

ITEMID: Un identificador único utilizado en la base de datos MIMIC-III para representar un concepto específico o medición registrada en el sistema.

CPT (Current Procedural Terminology): Un conjunto de códigos utilizados para describir procedimientos y servicios médicos con fines de facturación.

ICD (International Classification of Diseases): Un sistema de clasificación médica estándar utilizado para codificar diagnósticos y procedimientos.

CGID: Identificador único para cada cuidador registrado en la base de datos MIMIC-III.

CHARTEVENTS: Una tabla en la base de datos MIMIC-III que contiene observaciones clínicas registradas de los pacientes.

LABEVENTS: Una tabla en la base de datos MIMIC-III que contiene resultados de pruebas de laboratorio de los pacientes.

PRESCRIPTIONS: Una tabla en la base de datos MIMIC-III que contiene medicamentos ordenados para los pacientes.

D_ITEMS y D_LABITEMS: Tablas de diccionario en la base de datos MIMIC-III que proporcionan definiciones para los ITEMIDs utilizados en otras tablas.

API Rest: Interfaz de Programación de Aplicaciones que sigue los principios de la arquitectura REST (Transferencia de Estado Representacional), usada para la comunicación e integración entre sistemas en internet.

Python: Lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su fácil lectura y escritura de código.

Flask: Micro framework para Python utilizado para desarrollar aplicaciones web, incluyendo API web.

Controladores: En el contexto de una API, son los componentes que gestionan la entrada del usuario, trabajan con el modelo y, finalmente, devuelven una respuesta al cliente.

Repositorios: En la arquitectura de software, se refiere a objetos o lugares donde se almacenan, recuperan y buscan colecciones de datos.

Modelos: En programación, son estructuras de datos que representan las entidades del dominio y contienen la lógica de negocio relacionada con esos datos.

CITI Program: Programa educativo en línea que proporciona cursos de formación en temas como la investigación con seres humanos y el manejo de datos biológicos.

.Net: Plataforma de desarrollo de software de Microsoft para construir aplicaciones para web, Windows, Microsoft Azure, y más.

Driver de conexión: Software que permite a una aplicación interactuar con una base de datos, facilitando operaciones como la consulta y actualización de datos.

VPC (Virtual Private Cloud): Recurso de aislamiento en la nube que permite a los usuarios de AWS lanzar recursos en una red virtual definida por el usuario.

Parada cuidadosa (Graceful Stop): Proceso que permite que las operaciones activas finalicen de manera natural antes de que un sistema o aplicación se cierre.

Suplantación de identidad (Spoofing): Acto de disfrazarse como alguien más o falsificar datos con el objetivo de ganar acceso no autorizado o robar información.

Manipulación (Tampering): Acción de alterar o modificar maliciosamente datos o un sistema para dañar o obtener beneficios indebidos.

Repudio (Repudiation): Acto de negar haber llevado a cabo una acción o transacción, creando incertidumbre sobre la autenticidad de los eventos o registros.

Divulgación de Información (Information Disclosure): Exposición no autorizada de información privada o confidencial.

Denegación de Servicio (DoS): Ataque cibernético que intenta hacer que un sistema o red sea inaccesible para sus usuarios legítimos.

Autenticación Fuerte: Método de verificación de identidad que utiliza múltiples factores, incrementando la seguridad al requerir múltiples formas de validación.

Auditoría y Logging: Registro sistemático y revisión de eventos y cambios en un sistema para garantizar la seguridad y cumplimiento de normas.

Encriptación Robusta: Uso de algoritmos de cifrado fuertes y técnicas de seguridad para proteger los datos contra el acceso no autorizado.

Resistencia a DoS: Técnicas y medidas implementadas para proteger un sistema contra ataques de Denegación de Servicio.

Threat Modeling Tool: Herramienta de software utilizada para identificar y manejar posibles amenazas, evaluando el entorno y las vulnerabilidades de un sistema.

Ataque de secuestro de subdominio: Un tipo de ataque cibernético en el que un actor malicioso toma el control de un subdominio de una organización debido a configuraciones inadecuadas o registros DNS obsoletos.

Inyección SQL: Técnica de ataque en la que se inserta código malicioso en cadenas que son luego pasadas a una instancia de SQL Server para su análisis y ejecución.

Elevación de Privilegios: Técnica que permite que un usuario o proceso adquiera privilegios en un sistema informático, superiores a los que originalmente le fueron asignados.

Divulgación de Información: Exposición no intencionada de información confidencial debido a procesos de seguridad insuficientes o ataques maliciosos.

HTTPS (Protocolo Seguro de Transferencia de Hipertexto): Extensión de HTTP que se utiliza para la transmisión segura de datos, donde la comunicación es encriptada.

Repudio: En seguridad informática, se refiere a la capacidad de un usuario para negar haber realizado una acción que comprometa la seguridad, lo que hace difícil probar la autoría en el acto malicioso.

Suplantación de Identidad (Spoofing): Acción maliciosa donde un individuo o programa se hace pasar por otro para ganar acceso no autorizado o robar información.

Manipulación: En el contexto de la seguridad informática, se refiere al acto de alterar digitalmente los datos o sistemas para obtener un beneficio o causar daño.

Inyección SQL: Tipo de ataque de seguridad que explota vulnerabilidades en la capa de base de datos de una aplicación. El atacante introduce código malicioso en un campo de entrada.

DNS (Sistema de Nombres de Dominio): Sistema que traduce los nombres de dominio legibles por humanos en direcciones IP numéricas que identifican y localizan un recurso en una red.

CNAME (Canonical Name Record): Tipo de registro DNS que se utiliza para especificar que un nombre de dominio es un alias para otro dominio, el "nombre canónicoreal.

Secuestro de Subdominio: Ataque donde el atacante gana acceso y controla el subdominio de un sitio web, generalmente explotando un CNAME obsoleto o configuraciones de seguridad laxas.

Azure Traffic Manager: Servicio de balanceo de carga basado en DNS ofrecido por Microsoft Azure. Distribuye el tráfico de red de forma global a través de varios servicios en la nube.

Modelado de Amenazas: Proceso estructurado que se utiliza para identificar y evaluar los riesgos asociados con la seguridad de un sistema informático.

Command Query Responsibility Segregation (CQRS): Patrón de diseño de software que separa las operaciones de lectura y escritura en sistemas de información, permitiendo la optimización independiente y potencial escalabilidad de ambos procesos.

Patrones de software: Soluciones reutilizables a problemas comunes en el diseño de software, generalmente, ofrecen una estructura que ayuda a resolver problemas similares de diseño en diversas circunstancias.

Bibliografía

[1] I. H. Akin y B. Sunar, «On the Difficulty of Securing Web Applications Using CryptDB,» en *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, IEEE, dic. de 2014, págs. 745-752, ISBN: 978-1-4799-6719-3. DOI: 10.1109/BDCloud.2014.75.

[2] R. A. Popa, C. M. S. Redfield, N. Zeldovich y H. Balakrishnan, «CryptDB,» en *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11*, ACM Press, 2011. DOI: 10.1145/2043556.2043566.

[3] Institute of Electrical and Electronics Engineers, *On the Use of CryptDB for Securing Electronic Health Data in the Cloud: A Performance Study*, ISBN: 9781467383257.

[4] J. L. Fernández-Alemán, I. C. Señor, P. á. O. Lozoya y A. Toval, *Security and privacy in electronic health records: A systematic literature review*, jun. de 2013. DOI: 10.1016/j.jbi.2012.12.003.

[5] ORACLE. (jul. de 2020). «MySQL Enterprise Transparent Data Encryption (TDE).»

[6] Oracle. (jul. de 2020). «Securing Stored Data Using Transparent Data Encryption.»

[7] Microsoft. (jul. de 2020). «Transparent Data Encryption (TDE).»

[8] D. R. Q. Dr. Anwar Pasha Abdul Gafoor Deshmukh, «Transparent Data Encryption- Solution for Security of Database Contents,» en *International Journal of Advanced Computer Science and Applications*, ép. 3, vol. 2, mar. de 2011, págs. 25-28.

[9] F. Shahzad, W. Iqbal y F. S. Bokhari, «On the use of CryptDB for securing Electronic Health data in the cloud: A performance study,» en *2015 17th*

International Conference on E-health Networking, Application & Services (HealthCom), IEEE, oct. de 2015. DOI: 10.1109/healthcom.2015.7454484.

[10] ——, «On the use of CryptDB for securing Electronic Health data in the cloud: A performance study,» en *2015 17th International Conference on E-health Networking, Application & Services (HealthCom)*, IEEE, 2015, págs. 120-125.

[11] H. Yousuf, S. Salloum, A. Aburayya, M. Al-Emran y K. Shaalan, «A systematic review of CRYPTDB: Implementation, challenges, and future opportunities,» *Journal of Management Information and Decision Sciences*, vol. 24, págs. 1-16, 2021.

[12] Y. Zhuang, C. Wei, J. Li y W. Li, «Performance enhanced for CryptDB based on AES-NI acceleration,» en *Proc. 2nd Int. Conf. Advances Manage. Eng. Inf. Technol.*, 2017, págs. 357-361.

[13] L. Yao y X. Shuai, «Accelerate the paillier cryptosystem in CryptDB by Chinese remainder theorem,» en *2018 20th International Conference on Advanced Communication Technology (ICACT)*, IEEE, 2018, págs. 74-77.

[14] X. Jiang, X. Kong y Z. Xu, «Research on order-preserving encryption scheme based on CryptDB,» en *Journal of Physics: Conference Series*, IOP Publishing, vol. 1550, 2020, pág. 032 106.

[15] K. Foltz y W. R. Simpson, «Extending CryptDB to Operate an ERP System on Encrypted Data.,» en *ICEIS (1)*, 2018, págs. 103-110.

[16] R. A. Popa, N. Zeldovich y H. Balakrishnan, «Guidelines for using the CryptDB system securely,» *Cryptology ePrint Archive*, 2015.

[17] H. H. Nasereddin y A. J. Darwesh, «An object oriented programming on encrypted database system (CryptDB),» *Talent Development & Excellence*, vol. 12, n.º 1, págs. 5140-5146, 2020.

[18] J. Rauthan y K. Vaisla, «Scrambled database with encrypted query processing: CryptDB a computational analysis,» en *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, IEEE, 2017, págs. 199-211.

[19] M. Skiba, M. S. C. Mainka, D.-I. V. Mladenov y J. Schwenk, «Bachelor Thesis Analysis of Encrypted Databases with CryptDB,» 2015.

[20] A. Kumar y M. Hussain, «Secure query processing over encrypted database through cryptdb,» en *Recent Findings in Intelligent Computing Techniques: Proceedings of the 5th ICACNI 2017, Volume 3*, Springer, 2018, págs. 307-319.

[21] N. Aburawi, «Cryptdb mechanism on graph databases,» Tesis doct., University of Liverpool, 2020.

[22] M. Maisura, «Analyzing Client-side Encryption Implemented in Cryptdb,» *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, vol. 2, n.º 1, págs. 69-83, 2018.

[23] N. A. D. Lopez y R. B. L. Chua, «Securing Health Information System with CryptDB,» en *Theory and Practice of Computation: Proceedings of Workshop on Computation: Theory and Practice WCTP2015*, World Scientific, 2017, págs. 136-158.

[24] X. Yi, R. Paulet y E. Bertino, *Homomorphic Encryption and Applications*. Springer International Publishing, 2014. DOI: 10.1007/978-3-319-12229-8.

[25] M. L. Gaid y S. A. Salloum, «Homomorphic Encryption,» en *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2021)*, Springer International Publishing, 2021, págs. 634-642. DOI: 10.1007/978-3-030-76346-6_56.

[26] P. Paillier, «Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,» en *Advances in Cryptology — EUROCRYPT '99*, Springer Berlin Heidelberg, págs. 223-238. DOI: 10.1007/3-540-48910-x_16.

[27] C. Gentry, «Fully homomorphic encryption using ideal lattices,» en *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, ACM Press, 2009. DOI: 10.1145/1536414.1536440.

[28] Z. Brakerski y V. Vaikuntanathan, «Efficient Fully Homomorphic Encryption from (Standard) *maths f*{LWE},» *SIAM Journal on Computing*, vol. 43, n.º 2, págs. 831-871, ene. de 2014. DOI: 10.1137/120868669.

[29] D. X. Song, D. Wagner y A. Perrig, «Practical techniques for searches on encrypted data,» en *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, IEEE Comput. Soc. DOI: 10.1109/secpri.2000.848445.

[30] «Architecting for the Cloud AWS Best Practices,» inf. téc., 2018.

[31] V. R. Basili, R. W. Selby y D. H. Hutchens, «Experimentation in software engineering,» *IEEE Transactions on Software Engineering*, vol. SE-12, n.º 7, págs. 733-743, jul. de 1986. DOI: 10.1109/tse.1986.6312975.

[32] J. H. Hayes, «Energizing software engineering education through real-world projects as experimental studies,» en *Proceedings 15th Conference on Software Engineering Education and Training (CSEE&T 2002)*, IEEE Comput. Soc. DOI: 10.1109/csee.2002.995211.

[33] G. Succi, M. Stefanovic y W. Pedrycz, «Quantitative assessment of extreme programming practices,» en *Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No.01TH8555)*, IEEE. DOI: 10.1109/ccece.2001.933661.

[34] R. van Solingen y E. Berghout, «Integrating goal-oriented measurement in industrial software engineering: industrial experiences with and additions to the Goal/Question/Metric method (GQM),» en *Proceedings Seventh International Software Metrics Symposium*, IEEE Comput. Soc. DOI: 10.1109/metric.2001.915533.

[35] B. Kitchenham, «Procedures for performing systematic reviews,» *Keele, UK, Keele University*, vol. 33, n.º 2004, págs. 1-26, 2004.

[36] D. Moher, A. Liberati, J. Tetzlaff y D. G. Altman, «Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement,» *Public Library of Science Medicine*, vol. 6, n.º 7, e1000097, 2009.

[37] C. Collaboration, *Cochrane Handbook for Systematic Reviews*. Wiley, 2006.

[38] J. B. Institute, *JBI Reviewers' Manual*. The Joanna Briggs Institute, 2014.

[39] A. M. Cohill, D. M. Gilfoil y J. V. Pilitsis, «A Methodology for Evaluating Application Software,» inf. téc.

[40] H. Boral y D. J. Dewitt, «A Methodology for Database System Performance Evaluation,» inf. téc., 1984.

[41] L. Roberts y T. P. Moran, «The Evaluation of Text Editors: Methodology and Empirical Results,» inf. téc.

[42] R. Plosch, A. Dautovic y M. Saft, «The Value of Software Documentation Quality,» en *2014 14th International Conference on Quality Software*, IEEE, oct. de 2014. DOI: 10.1109/qsic.2014.22.

[43] Oracle, *14.8.5 Configuring Thread Concurrency for InnoDB*, Oracle, https://dev.mysql.com/doc/refman/5.6/en/innodb-performance-thread$_c$*oncurrency.html*.

[44] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan y N. Zeldovich, «Relational Cloud: A Database-as-a-Service for the Cloud,» inf. téc. dirección: https://dspace.mit.edu/handle/1721.1/62241.

[45] M. F. K. Stephen Tu, «Processing Analytical Queries over Encrypted Data,» *MIT CSAIL*, 2013.

[46] J. Lee, D. J. Scott, M. Villarroel, G. D. Clifford, M. Saeed y R. G. Mark, *Open-Access MIMIC-II Database for Intensive Care Research*. 2011, ISBN: 9781424441228. DOI: 10.0/Linux-x86{_}64.

[47] Flask, 2010.

[48] «Amazon Relational Database Service User Guide Amazon Relational Database Service: User Guide,» inf. téc., 2020.

[49] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[50] M. Howard y S. Lipner, *The security development lifecycle: SDL: a process for developing demonstrably more secure software*. Microsoft Press, 2006.

[51] J. Clarke, *SQL injection attacks and defense*. Elsevier, 2009.