ISSN: 3061-8991

Vol. 2 / 2026

Actas de Ciberseguridad para la Industria 5.0

Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

INSTITUTO NACIONAL DE ASTROFÍSICA, ÓPTICA Y ELECTRÓNICA

Actas de Ciberseguridad para la Industria 5.0

ISSN: 3061-8991

Actas de Ciberseguridad para la Industria 5.0 Vol. 2 (2026) es una publicación anual editada por el Instituto Nacional de Astrofísica Óptica y Electrónica, Calle Luis Enrique Erro, 1, Santa María Tonantzintla, San Andrés Cholula, C.P. 72840, Puebla, Pue., Tel. 2222663100, ext. 8303, página electrónica de la revista: https://actasciberseguridad.inaoep.mx/, Correo electrónico: actas.rci@inaoep.mx.
Editor responsable: Dr. Miguel Morales Sandoval. Reserva de Derechos al Uso Exclusivo No. 04-2025-070114531400-102, ISSN: 3061-8991, ambos otorgados por el Instituto Nacional del Derecho de Autor.
Responsable de la última actualización de este número, Lic. Liliana Perea Centeno, Instituto Nacional de Astrofísica Óptica y Electrónica, Calle Luis Enrique Erro, 1, Santa María Tonantzintla, San Andrés Cholula, C.P. 72840, Puebla, Pue., fecha de última modificación: Julio de 2026. Tamaño del archivo: 38MB.

Contenido

Editorial

En este segundo número de la revista *Actas de Ciberseguridad para la Industria 5.0* se presentan ocho artículos, donde se discuten problemáticas y avances en el área de ciberseguridad, desde la perspectiva de la investigación y la academia, con aplicaciones en la industria.

Los dos primeros artículos están centrados en la seguridad de imágenes y contenido multimedia. Marco Tulio Ramírez Torres et al., analizan un sistema de cifrado para imágenes, donde con estadística y ataques criptoanáliticos se ha podido verificar la calidad del cifrado, utilizando el ataque chosen plain image attack. El ataque permitió revelar la información original sin conocer la llave de cifrado, utilizando solo dos imágenes arbitrarias. En su contribución, los autores explican la forma de realizar un criptoanálisis, para que a futuro las personas que diseñen un sistema de seguridad contemplen estas pruebas. Por su parte, Alejandro Salinas Victorino et al., describen una arquitectura integral para la protección de imágenes digitales, mediante un esquema de inserción de marcas de agua en el dominio de la frecuencia utilizando la Transformada Discreta del Coseno (DCT) y fragmentación criptográfica mediante Shamir Secret Sharing (SSS) para garantizar confidencialidad en la distribución del contenido. Su propuesta incorpora un mecanismo de validación de integridad visual basado en hashes perceptuales, permitiendo comparar la similitud entre la marca de agua original y recuperada. Los resultados revelan que su esquema garantiza autenticidad, confidencialidad e integridad visual, resultando especialmente adecuada para entornos donde la confianza en el contenido digital es fundamental.

A continuación, contamos con cuatro contribuciones donde el cifrado se aplica en diversos contextos para brindar seguridad, tanto en datos en reposo como en tránsito. Salinas Victorino nos presenta una contribución en entornos del Internet de las Cosas (IoT). Se presenta una arquitectura de ejecución segura de firmware en tiempo de ejecución. Los módulos funcionales del firmware se almacenan cifrados en memoria flash y son validados mediante códigos de autenticación (HMAC) antes de su ejecución dinámica. Esta arquitectura previene la exposición de datos sensibles y refuerza la integridad y autenticidad del firmware, incluso ante ataques con acceso físico al dispositivo. Cecilia Angulo Domínguez et al., abordan el problema de la delegación de la custodia de los datos a terceros, en el contexto del almacenamiento en la nube donde es muy probable que los datos almacenados puedan ser alterados o eliminados sin conocimiento del propietario, lo cual compromete no solo la disponibilidad, sino también la veracidad de la información. Para abordar este problema, los autores presentan un esquema Provable Data Possession (PDP), el cual permite verificar que los datos almacenados en un servidor remoto se mantienen íntegros sin necesidad de descargarlos en su totalidad. Se presenta una arquitectura escalable del esquema PDP dentro de una infraestructura basada en servicios de Amazon Web Services (AWS), basada en tres componentes principales: el cliente, el servidor y el verificador. Ricardo A. Ibarra García et al., presenta un modelo dinámico de control de acceso basado en eventos, donde un evento se define como una conjunción de atributos: sujeto, recurso, espacio, tiempo y acción. El modelo se evaluó a través de un prototipo funcional en un estudio de caso simulado, utilizando políticas construidas a partir del dataset KDD Cup 1999. Shanel Reyes Palacios et al., por su parte, nos presentan el diseño y la arquitectura de un servicio de cifrado homomórfico, validado con los esquemas mayormente usados en Minería de Datos con

Preservación de la Privacidad: Liu, CKKS y Paillier. Una ventaja del modelo es que puede soportar cualquier otro esquema de cifrado, incorpora patrones de paralelismo y esquemas de distribución de carga de trabajo.
Finalmente, contamos con dos contribuciones enfocadas a la detección de ataques. José de Jesús Zapata Lara describe los resultados de la construcción de un clúster para el entrenamiento de modelos basados en Machine Learning (ML), orientados a la detección de ataques de negación de servicio distribuidos (DDoS) utilizando el sistema de almacenamiento de archivos distribuido HDFS del framework Hadoop y la integración del framework Spark para procesamiento. Por su parte, Stanislav Glebskiy et al., discuten de manera general sobre las tendencias en el desarrollo de métodos para detectar y prevenir ataques iSQL. Su contribución pretende servir de introducción a cualquier persona que quiera adentrarse en el mundo de las iSQL (inyecciones SQL) y su mitigación, presentando un panorama general del área y discutiendo los enfoques más recientes de investigación.

Todos los trabajos previos se presentaron y discutieron durante la 4ta Reunión de Ciberseguridad para la Industria 5.0, evento en el participaron más de 160 personas de más de 20 organizaciones, entre ellas empresas privadas, centros de investigación universidades e instituciones públicas. Todos los artículos fueron dictaminados por el comité editorial y revisores expertos en el área.

Criptoanálisis a Sistema de Cifrado de Imágenes Basado en Múltiples Cajas de Sustitución

Marco Tulio Ramírez-Torres
Coordinación Académica Región Altiplano Oeste
Universidad Autónoma de San Luis Potosí
Salinas de Hidalgo, México
tulio.torres@uaslp.mx

Gina Gallegos-García
Centro de Investigación en Computación
Instituto Politécnico Nacional
Ciudad de México, México
ggallegos@cic.ipn.mx

Jesús Agustín Aboyte-González
Centro de Investigación en Computación
Instituto Politécnico Nacional
Ciudad de México, México
agustin.aboytes@upslp.edu.mx

César Arturo Guerra-García
Coordinación Académica Región Altiplano Oeste
Universidad Autónoma de San Luis Potosí
Salinas de Hidalgo, México
cesar.guerra@uaslp.mx

Carlos Soubervielle-Montalvo
Facultad de Ingeniería
Universidad Autónoma de San Luis Potosí
San Luis Potosí, México
carlos.soubervielle@uaslp.mx

***Abstract*— Debido a las necesidades actuales de seguridad, diversos autores han propuesto varios sistemas de cifrado, basados en diferentes principios de funcionalidad y aplicados a diferentes tipos de información. Las cajas de sustitución han demostrado ser una herramienta importante en los sistemas de cifrado simétrico como parte de las operaciones de confusión. Actualmente hay propuestas de sistemas de cifrado que incluyen el uso de múltiples cajas de sustitución, así como las llamadas cajas de sustitución dinámicas. Sin embargo, es importante analizar a profundidad los sistemas de seguridad que se encuentran publicados antes de ser implementados o tomados como referencia. En este trabajo se analiza un sistema de cifrado para imágenes, propuesto por Majid Khan, donde con estadística y ataques criptoanáliticos se ha podido verificar la calidad del cifrado, utilizando el ataque chosen-plain image attack. El sistema es quebrado con este ataque, revelando la información original sin conocer la llave de cifrado, utilizando solo dos imágenes arbitrarias. Este trabajo trata de explicar a detalle la forma de realizar un criptoanálisis, para que a futuro las personas que diseñen un sistema de seguridad contemplen estas pruebas.**

Keywords—criptoanálisis, caja de sustitución, cifrado de imágenes.

I. Introducción

En la actualidad, debido a la demanda de seguridad de todos los procesos que requieren trabajar en línea, y el almacenamiento seguro de información confidencial, se han propuesto diversos algoritmos criptográficos con diferentes enfoques, entre ellos los de enfoque caótico. Esto debido a propiedades que tienen como ergodicidad y la sensibilidad a condiciones iniciales.

Por otra parte, el cifrado de contenido multimedia ha requerido el desarrollo de nuevos sistemas de cifrado, debido a que esta información requiere seguridad criptográfica y seguridad perceptual [1]. Para explicar esto, un ejemplo pueden ser las imágenes digitales, las cuales pueden contener una gran cantidad de datos, una alta redundancia y correlación adyacente. Para cifrar una imagen no solo se requiere proteger la información modificando su contenido de una forma segura (seguridad criptográfica), si no que la versión cifrada debe evitar revelar patrones de la imagen original (seguridad perceptual). Un ejemplo de cuando no se considera la seguridad perceptual se puede ver en la Fig. 1.

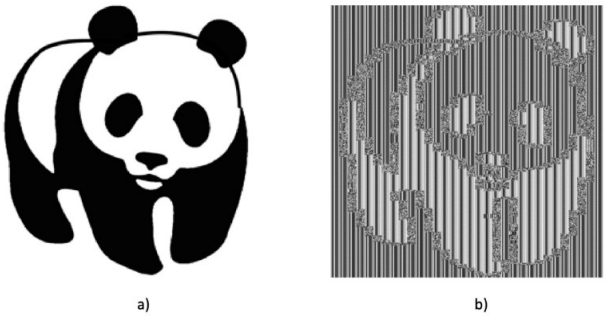


Fig. 1. a) Imagen original de dos tonos, b) versión cifrada de a) con el sistema AES en modo ECB.

Sin embargo, en muchas ocasiones estos nuevos sistemas presentan debilidades ante ataques de criptoanálisis y/o criptoanálisis diferencial. Provocando la fuga de información confidencial. Por ejemplo, en [2] utilizan una secuencia pseudoaleatoria generada por un sistema hipercaótico para cifrar las imágenes, utilizando la operación XOR y la suma modular. Este sistema fue analizado en el trabajo [3] y se encontraron debilidades al momento de aplicar el ataque Chosen-Plain Image

Attack (CPIA). Por otra parte, en [4] se propone un método de cifrado y de generación de cajas de sustitución basado en caos. Sin embargo, en 2018 en [5], Zhu junto con otros autores publicaron las debilidades del sistema propuesto por Çavuşoğlu et. al, bajo el ataque Chosen-Plaintext Attack (CPA). En [6] los autores proponen un algoritmo de cifrado de imágenes caótico, basado en la entropía de la información. En el mismo año, Li et. al, en [7] revelan los problemas de seguridad que presenta dicho sistema contra ataques diferenciales. Por lo que se puede ver, es necesario un análisis más profundo con diferentes escenarios para poder validar un nuevo sistema de cifrado.

En este trabajo se analiza una propuesta hecha por Majid Khan en [8], la cual presenta una debilidad ante el ataque CPIA. El sistema utiliza los parámetros iniciales de un sistema caótico como llave secreta para generar cajas de sustitución. Estas cajas son también llamadas S-box, en este sistema las emplean para cifrar imágenes sustituyendo los valores de los coeficientes de los píxeles. En el artículo original presentan una validación del sistema, principalmente basada en pruebas estadísticas y con poca variedad de imágenes. Por lo que, en esta investigación se busca describir a detalle el ataque, su justificación y referencia, para que en futuras propuestas contemplen una mayor variedad de pruebas de forma justificada.

Este artículo tiene la siguiente estructura, en la sección 2 se describe el sistema propuesto por Khan, tanto la generación de las cajas de sustitución como el algoritmo de cifrado. La sección 3 muestra el criptoanálisis aplicado paso a paso. Y finalmente, en la sección 4 se incluyen las conclusiones.

II. Descripción del sistema de cifrado

El sistema de cifrado propuesto por Khan [8] utiliza sistemas caóticos que suelen ser empleados en criptografía debido a su ergodicidad y a que son sensibles a condiciones iniciales. Dicho esquema está basado en un sistema caótico multi-paramétrico, utilizando la combinación de los sistemas de Lorenz y Rossler. El autor afirma que este sistema es altamente no lineal y tiene comportamiento caótico. El conjunto de ecuaciones de este sistema se muestra a continuación:

$$\frac{dx}{dt} = \sigma(y - x) - y - z,$$

$$\frac{dy}{dt} = rx - y - 20xz + x + ay, \qquad (1)$$

$$\frac{dz}{dx} = 5x - bz + \delta + x(x - c).$$

Para el sistema de ecuaciones (1) se establecen los parámetros δ, r, a, b y c que son los que gobiernan las características de salida y los valores generados por el sistema.

El algoritmo de cifrado de imágenes es descrito a continuación:

1. Las trayectorias son generadas en el sistema caótico multi-paramétrico, seleccionando las condiciones iniciales.

2. Se colectan las muestras de la trayectoria de datos caóticos.

3. Aplicando una transformación afín, las muestras de la salida son ajustadas en un rango de 0 a 255.

4. Se conforma la S-box como una matriz de 16×16, utilizando las muestras ajustadas en el paso anterior.

5. Los píxeles de las imágenes que serán los datos de entrada a la S-box, son convertidos a números binarios de ocho bits.

6. Dicho número binario debe ser separado en dos bloques de cuatro bits, también llamados nibbles y convertidos nuevamente a base decimal.

7. La imagen cifrada se obtiene al usar los dos números (de cuatro bits) del paso 6 provenientes del coeficiente del pixel. Estos dos números serán usados como coordenadas de reglón y columna en la S-box. El número de los cuatro bits más significativos señala en la S-box, el renglón del valor de sustitución. Mientras que el número de los cuatro bits menos significativos señala la columna. El número localizado en la S-box se utiliza para sustituir el coeficiente del pixel en la imagen original. Este paso se repite hasta cifrar toda la imagen.

Para explicar con más detalle el paso 7 del algoritmo, suponga el caso de un píxel cuyo coeficiente es 21 (hex). En la Fig. 2 se puede ver una sección de la S-box del sistema AES. Los bits más significativos del valor de entrada 21 (hex), señalan el renglón 2, mientras que los menos significativos la columna 1. Por lo tanto, el valor 21 (hex) se sustituye por fd (hex).

Estos son todos los pasos que abarca el esquema de cifrado diseñado por Khan. Consta únicamente de una función de confusión que intercambia los valores de entrada (los coeficientes de píxel) por los valores de la caja de sustitución. El resto del artículo original lo dedica a validar el cifrado con pruebas estadísticas y criptoanálisis diferencial.

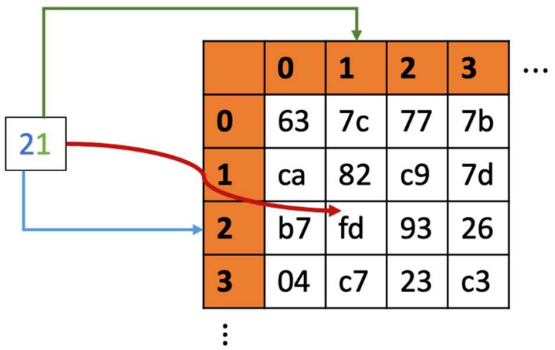


Fig. 2. Sustitución del valor 21 (hex) utilizando la caja de sustitución del sistema AES.

III. Criptoanálisis propuesto

Antes de comenzar a describir el ataque que se emplea en esta investigación, es importante definir lo que es una debilidad o ruptura en un sistema de cifrado. En [9] el autor define romper un método de cifrado, como encontrar una debilidad que puede ser explotada con menor complejidad que por fuerza bruta. Por lo tanto, es necesario aclarar que quizás se requieran cantidades irreales de conjuntos de textos conocidos o escogidos por el

atacante. Así como considerar que el adversario tiene habilidades que tal vez no sean posibles en el mundo real.

El ataque utilizado en este criptoanálisis es el Chosen Plain-Image Attack (CPIA), es una versión específica para imágenes del ataque Chosen Plaintext Attack, que ha sido utilizado en otros criptoanálisis, por ejemplo en [10] y [11]. En este ataque el adversario es capaz de escoger las imágenes en claro a la entrada del sistema, y obtener las respectivas imágenes cifradas. En este ataque el adversario tratará de encontrar alguna debilidad en el sistema sin conocer la llave secreta, utilizando únicamente las imágenes en claro y cifradas.

Debido a que es parte del propósito de este artículo servir como una explicación detallada de las pruebas de criptoanálisis, es conveniente aclarar que una vez especificadas las consideraciones del ataque, la forma en que se llevará a cabo no es única. Se pueden hacer diferentes usos con la información disponible, diferentes pruebas y análisis. Así como la selección de las imágenes en claro pueden tener diferentes propósitos. Por lo tanto, repetir un ataque tal cual como es reportado en algún artículo, para un nuevo sistema puede no ser concluyente para medir la calidad del cifrado. Antes de replicar algo, se debe analizar el porqué de las condiciones y si cumplen con los supuestos del ataque.

Una vez explicado esto, se procede a realizar el ataque. Las imágenes seleccionadas por el atacante son de 8 bits, de 256*256 pixeles y se muestran en la Fig. 3.

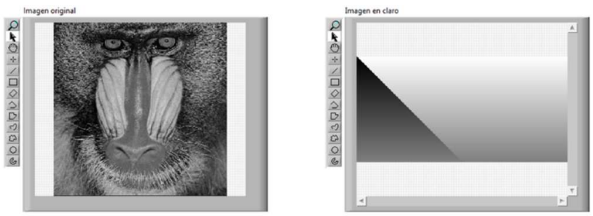


Fig. 3. Imágenes en claro. a) Imagen de mandril, ampliamente utilizada en el procesamiento de imágenes. b) Imagen hecha con todos los colores de la escala de grises de 8 bits.

La Fig. 3b) es en escala de grises a 8 bits (0 a 255), fue diseñada para aportar dos tipos de información en este ataque. Se sabe la localización y cantidad de píxeles con un valor de coeficiente específico. Es decir, cada color se repite un número único de veces. La imagen tiene dimensiones de 128×257, bajo el siguiente patrón:

$$\begin{bmatrix} 0 & & 255 & 255 \cdots & & 255 \\ 1 & & 1 & 254 \cdots & & 254 \\ \vdots & & & \ddots & & \vdots \\ 127 & 127 & \ldots & 128 & \ldots & 128 \end{bmatrix} \quad (2)$$

Donde el primer renglón está formado por un píxel con el valor de 0 y 256 píxeles con el valor de 255. El segundo renglón contiene dos píxeles con el valor de 1 y 255 píxeles con el valor de 254. Así sucesivamente en cada renglón, un lado aumenta el número de píxeles de un valor, mientras que el número de píxeles del otro lado disminuye. El histograma es un gráfico para representar distribuciones de frecuencias de los valores de los píxeles, hecho con los valores de la Fig. 3b), el cual se ve de la siguiente manera:

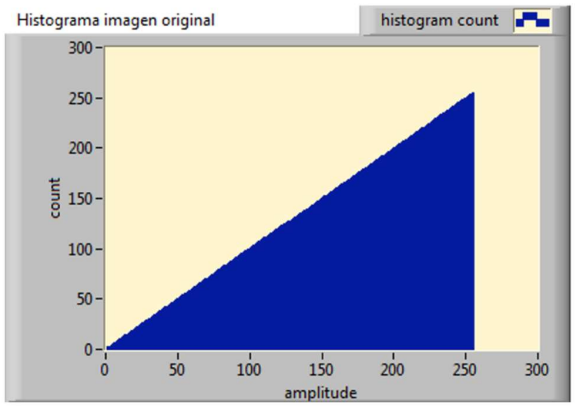


Fig. 4. a) Histograma de los coeficientes de los píxeles de la Fig. 3b).

Se pudo haber utilizado una imagen más sencilla, como la de la Fig. 5, que es igual a la imagen seleccionada en el criptoanálisis [12]. Es un barrido de todos los valores de 0 a 255. Sin embargo, pensando en más casos, nuestra propuesta serviría para sistemas de cifrado similares, incluso si estos tuvieran una operación de permutación. Las operaciones de permutación sirven para cambiar de posición a los píxeles, como por ejemplo las funciones de mapeo. Si el sistema propuesto por Khan o el sistema propuesto en [13], agregaran alguna función de mapeo, el criptoanálisis propuesto en [12] con la Fig. 5, no funcionaría ya que los pixeles se repiten en la misma frecuencia.

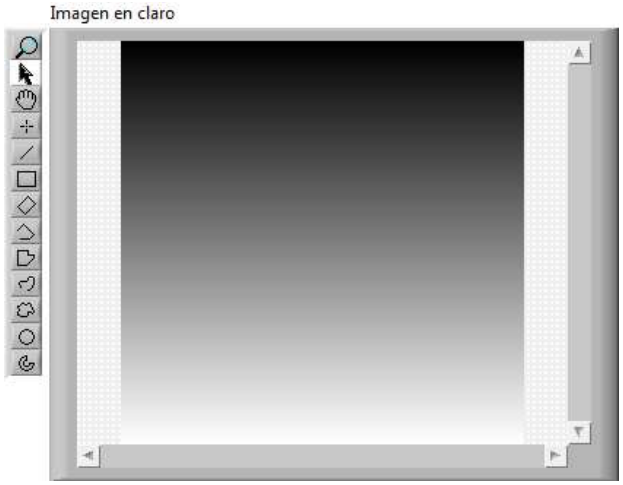


Fig. 5. Imagen en escala de grises de 256*256 pixeles, hecha con los valores de 0 a 255, un valor diferente por cada renglón.

Siguiendo con el ataque se seleccionan los parámetros iniciales del sistema caótico para poder generar la caja de sustitución. Para una mejor descripción se utilizará la caja de

sustitución reportada en el artículo por el autor, la cual se muestra a continuación

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	74	3	93	16	37	8B	5B	E1	A2	94	F2	14	77	2A	2B	F9
1	7D	C	9C	1C	32	7C	25	C2	A8	9F	D4	1A	7F	B	F7	52
2	85	15	A5	22	2D	6E	13	B9	AE	AA	D2	34	8F	6	72	66
3	8E	1E	AD	2C	20	5F	1	B0	B4	AF	D0	3A	90	23	69	DC
4	97	27	B6	31	19	4F	EF	A7	C1	BD	C9	48	89	33	58	9D
5	A0	30	BE	35	11	3F	DD	8C	CF	CA	C0	56	87	64	50	7E
6	A9	39	C6	3D	0	2F	CC	86	E4	D1	B7	5D	62	96	F	45
7	B2	43	CE	41	F6	1F	BA	81	F3	D5	B5	65	5C	FB	51	26
8	BB	4C	DE	44	EC	E	88	76	18	D8	B1	91	4E	D	6B	7A
9	C4	55	E6	46	E2	FD	78	73	2E	DB	AC	B8	47	17	95	FF
A	CD	5E	ED	49	D7	EB	68	71	36	E3	AB	4	FE	29	F5	63
B	D6	67	F4	4A	CB	DA	59	6F	53	E5	A3	21	D3	9A	9B	83
C	DF	70	FC	4D	BF	C8	3B	6D	5A	E7	A1	28	BC	57	D9	24
D	E8	79	2	4B	B3	A4	12	6C	61	E9	C3	42	98	38	F3	1D
E	F1	82	9	40	A6	92	5	84	75	EE	C7	54	1B	A	E0	9E
F	FA	8A	10	3C	99	80	F8	8D	7B	F0	8	6A	3E	60	C5	7

Fig. 6. Caja de sustitución reportada en [8].

Ambas imágenes en claro son cifradas con la misma caja de sustitución. El resultado se muestra en la Fig. 7.

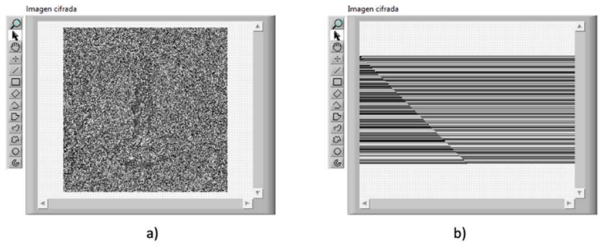


Fig. 7. Imágenes cifradas. a) Imagen del mandril cifrada. b) Imagen 3b) cifrada.

Si además analizamos el histograma de la Fig. 7b), obtendríamos la siguiente gráfica:

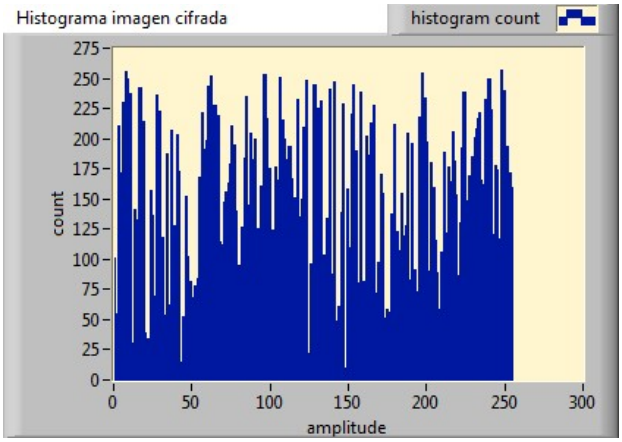


Fig. 8. Histograma de la Fig. 7b).

Como se puede ver, el sistema de cifrado no es capaz de ocultar la redundancia de la imagen original, solo intercambia los valores. Un buen sistema de cifrado arrojaría un histograma uniforme, sin importar la imagen de entrada. Esta debilidad nos permite romper el sistema, aquí se analizan dos formas, donde se podrían calcular una caja de sustitución inversa. La primera forma 1) Acorde a su posición en la imagen original. La Fig. 3b) sirve como referencia para saber la posición original de cada valor, mientras que en la imagen cifrada, Fig. 7b), permite analizar los nuevos valores de los píxeles, y acorde a su posición deducir el valor por el cual se intercambiaron los números de entrada 0, 1, 2, hasta 255. La segunda forma, 2) Haciendo un análisis de histograma. Utilizando el histograma de la imagen cifrada Fig. 7b) es posible deducir los valores por los cuales se intercambiaron los coeficientes de píxel. Esto debido a que se conoce el número de píxeles de cada valor en la imagen original, Fig. 3b). Por ejemplo, el valor que solo tenga un píxel originalmente era el valor de 0, el valor que tenga 2 píxeles era el valor 1 y así sucesivamente.

Utilizando las imágenes de las Fig. 3b) y 7b), y en caso de que se requiera, el histograma de la Fig. 8, es posible recuperar la Fig. 3a) a partir de la Fig. 7a) con un porcentaje del 100% de igualdad, sin conocer la llave de cifrado. El resultado se puede ver en la Fig. 9.

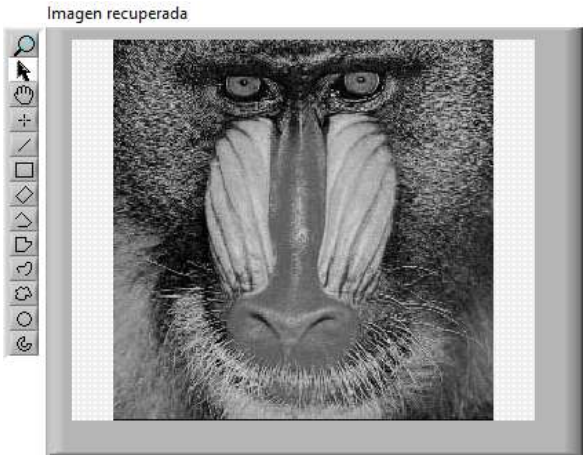


Fig. 9. Imágen recuperada de la Fig. 7b).

Como se puede ver, la imagen original del mandril se puede recuperar sin conocer la llave secreta. Se necesita solo una imagen conocida que contenga todos los valores de la codificación y se conozcan sus posiciones. Hay más ataques y vectores que podrían quebrar este sistema, por ejemplo si continúa con la Fig. 5, en la Fig. 10 podemos ver su versión cifrada. Esta imagen permite calcular la S-box inversa ya que la información no cambia de lugar, solo se sustituyen los valores. Por lo tanto el valor de renglón revela el valor original.

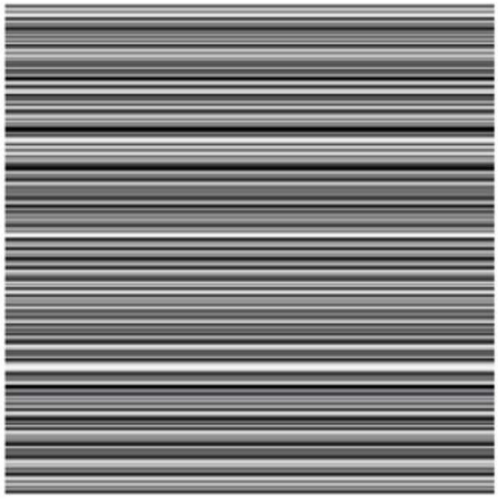


Fig. 10. Versión cifrada de la imagen 3b).

Otro vector de ataque es introducir 256 imágenes sólidas, es decir de un solo color, y con esa información calcular la S-box inversa. Incluso el sistema de cifrado también sería vulnerable al ataque Chosen Ciphertext Attack, al pasar la imagen de la Fig. 3b) por la operación de descifrado.

La condición de que ambas imágenes se cifraron con la misma caja de sustitución, o dicho de forma más general, que ambas imágenes fueron cifradas con la misma condición inicial, son parte de los aspectos que, si el sistema de cifrado no limita en sus pasos es válido aplicar de esta manera en el criptoanálisis. Como se mencionó anteriormente, en los criptoanálisis las reglas son un poco más flexibles, permitiendo considerar que el adversario puede hacer este tipo de acciones de cifrar con la misma condición inicial, como seleccionar las imágenes en claro, etc. Prueba de la utilidad de esto, es que los sistemas de cifrado avanzados pueden pasar estas pruebas, bajo las mismas consideraciones en los ataques.

IV. Conclusiones

En esta investigación se presenta un criptoanálisis a un sistema de cifrado para imágenes, basado en la generación de múltiples de cajas de sustitución a partir de un sistema caótico. El sistema de cifrado presenta una debilidad ante el ataque Chosen Plain-Image Attack, lo que implicaría que presenta debilidades ante otro tipo de ataques. Si el sistema hubiera sido evaluado con imágenes más simples, no hubiera pasado las pruebas estadísticas que reporta el autor en el artículo, desde la inspección visual. Otro problema que se puede señalar es que las cajas de sustitución generadas por el sistema caótico no son evaluadas. Cuando se propone un método de generación de cajas de sustitución, éstas se deben pasar por diversas pruebas, para confirmar que las cajas cumplen con ciertas propiedades. En la ref. [14] se puede consultar un trabajo de esa área donde se propone un método y se explican las pruebas a las que se someten las cajas.

Si se analizan los sistemas de cifrado que están bien establecidos como el AES (Advanced Encryption Standard), se puede ver que utilizan varias rondas con operaciones de confusión y difusión, incluyendo cajas de sustitución. Por lo tanto, la recomendación en el caso de diseño de sistemas de cifrado de imágenes, es hacer sistemas más robustos y combinar diversas operaciones para poder romper la alta correlación y la correlación adyacente.

Otra recomendación que se puede dar con este trabajo es que, al momento de diseñar sistemas de cifrado, se usen imágenes solidas que son el caso más simple, porque son de un solo color Si el sistema es bueno, será capaz de ocultar la alta redundancia de las imágenes en el cifrado. Esto se puede analizar calculando los histogramas.

Esperamos que este trabajo sirva de referencia para diseñadores de criptosistemas, y de esta manera contemplen aspectos de seguridad necesarios para poder pasar pruebas y ataques. Es muy importante evitar simplemente repetir punto a punto pruebas que se encuentran en la literatura, sin interpretar su porqué y si es aplicable al nuevo sistema que se esté diseñando.

References

[1] S. Lian, *Multimedia Content Encryption: Techniques and Applications*. Boca Raton, FL: CRC Press, 2008.

[2] C. Zhu, "A novel image encryption scheme based on improved hyperchaotic sequences," *Optics Communications*, vol. 285, no. 1, pp. 29–37, Jan. 2012.

[3] C. Li, Y. Liu, T. Xie, and M. Z. Chen, "Breaking a novel image encryption scheme based on improved hyperchaotic sequences," *Nonlinear Dynamics*, vol. 73, no. 3, pp. 2083–2089, Feb. 2013.

[4] Ü. Çavuşoğlu, S. Kaçar, I. Pehlivan, and A. Zengin, "Secure image encryption algorithm design using a novel chaos based S-Box," *Chaos, Solitons & Fractals*, vol. 95, pp. 92–101, Feb. 2017.

[5] C. Zhu, G. Wang, and K. Sun, "Cryptanalysis and improvement on an image encryption algorithm design using a novel chaos based S-box," *Symmetry*, vol. 10, no. 9, p. 399, Sep. 2018.

[6] G. Ye, C. Pan, X. Huang, Z. Zhao, and J. He, "A chaotic image encryption algorithm based on information entropy," *International Journal of Bifurcation and Chaos*, vol. 28, no. 01, p. 1850010, Jan. 2018.

[7] C. Li, D. Lin, B. Feng, J. Lü, and F. Hao, "Cryptanalysis of a chaotic image encryption algorithm based on information entropy," *IEEE Access*, vol. 6, pp. 75834–75842, 2018.

[8] M. Khan, "A novel image encryption scheme based on multiple chaotic S-boxes," *Nonlinear Dynamics*, vol. 82, no. 1, pp. 527–533, Apr. 2015.

[9] B. Schneier, "A self-study course in block-cipher cryptanalysis," *Cryptologia*, vol. 24, no. 1, pp. 18–33, Jan. 2000.

[10] Y. Chen, C. Tang, and R. Ye, "Cryptanalysis and improvement of medical image encryption using high-speed scrambling and pixel adaptive diffusion," *Signal Processing*, vol. 167, p. 107286, Jan. 2020.

[11] C. Li, S. Li, G. Chen, and W. A. Halang, "Cryptanalysis of an image encryption scheme based on a compound chaotic sequence," *Image and Vision Computing*, vol. 27, no. 8, pp. 1035–1039, Jul. 2009.

[12] M. Ahmad, M. N. Doja, and M. S. Beg, "Cryptanalysis and improvement of an image encryption scheme using Fourier series," *3D Research*, vol. 8, no. 4, pp. 1–11, Dec. 2017.

[13] M. Khan, "A novel image encryption using Fourier series," *Journal of Vibration and Control*, vol. 21, no. 16, pp. 3450–3455, Nov. 2015.

[14] J. A. Aboytes-González, J. S. Murguía, M. Mejía-Carlos, H. González-Aguilar, and M. T. Ramírez-Torres, "Design of a strong S-box based on a matrix approach," *Nonlinear Dynamics*, vol. 94, no. 3, pp. 2003–2012, Nov. 2018.

Esquema de Protección de Imágenes Digitales con Marca de Agua en DCT y Shamir Secret Sharing

Alejandro Salinas Victorino
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
alejandro.salinasv@inaoe.mx

Kelsey Alejandra Ramírez Gutiérrez
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
kramirez@inaoep.mx

***Resumen*—Este trabajo presenta una arquitectura integral para la protección de imágenes digitales que combina técnicas de esteganografía, criptografía y verificación visual. Se propone un esquema de inserción de marcas de agua en el dominio de la frecuencia utilizando la Transformada Discreta del Coseno (DCT) y fragmentación criptográfica mediante Shamir Secret Sharing (SSS) para garantizar confidencialidad en la distribución del contenido. Adicionalmente, se incorpora un mecanismo de validación de integridad visual basado en hashes perceptuales, permitiendo comparar la similitud entre la marca de agua original y recuperada. Los resultados experimentales evidencian una inserción de marca de agua imperceptible con PSNR $>$ 37 dB, una recuperación parcial confiable con $k = 3$ de $n = 6$ acciones del esquema de compartición, y una verificación visual efectiva con un nivel de similitud del 94 %. La solución propuesta garantiza autenticidad, confidencialidad e integridad visual, resultando especialmente adecuada para entornos donde la confianza en el contenido digital es fundamental.**

***Palabras clave*—Marca de agua digital, Transformada Discreta del Coseno (DCT), Shamir Secret Sharing, hash perceptual, distancia de Hamming, integridad visual, protección de imágenes.**

I. Introducción

En la actualidad, la protección de contenido visual digital se ha convertido en un desafío debido a la facilidad con la que las imágenes pueden ser manipuladas, copiadas y distribuidas en entornos digitales. Esta problemática impacta sectores sensibles como la medicina, la propiedad intelectual y los medios digitales, donde es crucial preservar la autenticidad, confidencialidad e integridad visual.

Las imágenes digitales, por su naturaleza fácilmente replicable y modificable, son particularmente vulnerables a amenazas como la falsificación, la distribución no autorizada y la manipulación maliciosa. Esta situación plantea la necesidad de mecanismos avanzados que integren técnicas robustas de ocultamiento de información, control de acceso al contenido y verificación de integridad, incluso tras procesos potencialmente inseguros de distribución o almacenamiento.

Una solución ampliamente adoptada para la protección de imágenes son las marcas de agua digitales, que insertan información relacionada con la autenticidad o propiedad intelectual de la imagen, sin afectar perceptiblemente su apariencia. Estas marcas pueden ser visibles o invisibles, frágiles o robustas, y pueden insertarse tanto en el dominio espacial como en el dominio de la frecuencia.

En este trabajo se propone un esquema de protección para imágenes digitales que combina la inserción de una marca de agua invisible y robusta en el dominio de la frecuencia mediante la Transformada Discreta del Coseno (DCT), con mecanismos de confidencialidad basados en el esquema de Shamir Secret Sharing (SSS) y verificación de integridad visual mediante hashes perceptuales.

La combinación de estos métodos permite abordar múltiples amenazas en escenarios reales: desde asegurar la autenticidad de diagnósticos médicos transmitidos entre instituciones, hasta la protección de imágenes sensibles en redes distribuidas o la validación de contenido multimedia en sistemas de vigilancia.

En resumen, el presente trabajo propone una arquitectura integral para la protección de imágenes digitales que combina inserción robusta de marca de agua en el dominio DCT, compartición segura mediante Shamir Secret Sharing y validación de integridad visual con hashes perceptuales. Los principales aportes de esta investigación son: (1) el diseño e implementación de un esquema híbrido que proporciona autenticidad, confidencialidad e integridad visual; y (2) la evaluación experimental de su eficacia en términos de imperceptibilidad, reconstrucción parcial y verificación visual. El artículo se organiza de la siguiente manera: en la Sección II se presenta el trabajo relacionado; en la Sección III se exponen los fundamentos teóricos; la Sección IV describe la metodología propuesta; en la Sección V se discuten los resultados obtenidos; finalmente, las Secciones VI y VII abordan las conclusiones y posibles líneas de trabajo futuro.

II. Trabajo relacionado

El diseño de esquemas de protección de imágenes digitales ha sido ampliamente abordado en la literatura mediante técnicas que buscan garantizar propiedades clave como la autenticidad, la confidencialidad y la integridad visual. En esta sección se presentan los trabajos más representativos que abordan los componentes esenciales de la propuesta aquí desarrollada.

En primer lugar, la Transformada Discreta del Coseno (DCT) ha sido empleada como una técnica robusta para la inserción de marcas de agua, gracias a su capacidad para resistir compresión con pérdida, ruido y manipulaciones moderadas. Srivastava y Garg [5] demuestran esta utilidad en un esquema

donde cada canal RGB de una imagen es transformado con DCT, se inserta la marca de agua en los coeficientes de frecuencia y luego se reconstruye mediante IDCT. El enfoque mantiene una alta calidad visual, evaluada mediante métricas como PSNR y MSE. No obstante, el trabajo carece de mecanismos adicionales de seguridad como validación de integridad visual o confidencialidad, aspectos que el presente proyecto sí aborda mediante perceptual hashing y Shamir Secret Sharing.

Por otra parte, la técnica de inserción implementada en este trabajo se fundamenta en el algoritmo propuesto por Katzenbeisser y Petitcolas [2], ampliamente citado en la literatura. Este método consiste en dividir la imagen en bloques de 8×8 píxeles, aplicar DCT a cada bloque, modificar coeficientes de frecuencia media según los bits de la marca de agua, y finalmente reconstruir la imagen mediante IDCT. Los algoritmos 1 y 2 ilustran los procesos de codificación y decodificación. Este enfoque ofrece un equilibrio adecuado entre invisibilidad y robustez frente a alteraciones comunes en entornos digitales.

Algoritmo 1 Proceso de codificación DCT-Steg

for $i = 1$, ..., $l(M)$ **do**
  Elegir un bloque de portadora b_i
  $B_i = \mathrm{D}\{b_i\}$ (Transformada DCT)
  if $m_i = 0$ **then**
    if $B_i(u_1, v_1) > B_i(u_2, v_2)$ **then**
      Intercambiar $B_i(u_1, v_1)$ y $B_i(u_2, v_2)$
    end if
  else
    if $B_i(u_1, v_1) < B_i(u_2, v_2)$ **then**
      Intercambiar $B_i(u_1, v_1)$ y $B_i(u_2, v_2)$
    end if
  end if
  Ajustar los valores para asegurar $|B_i(u_1, v_1) - B_i(u_2, v_2)| > x$
  $b'_i = \mathrm{D}^{-1}\{B_i\}$ (Transformada Inversa IDCT)
end for
Construir la imagen esteganográfica combinando todos los bloques b'_i

Algoritmo 2 Proceso de decodificación DCT-Steg

for $i = 1$, ..., $l(M)$ **do**
  Obtener el bloque de portadora b_i correspondiente
  $B_i = \mathrm{D}\{b_i\}$ (Transformada DCT)
  if $B_i(u_1, v_1) \leq B_i(u_2, v_2)$ **then**
    $m_i = 0$
  else
    $m_i = 1$
  end if
end for

En lo que respecta a confidencialidad y control de acceso, el esquema criptográfico de *Shamir Secret Sharing* (SSS) ha sido incorporado en distintos contextos de seguridad visual. Un ejemplo relevante es el trabajo de Kaçamak y Uka [3], quienes propusieron una estrategia de ocultamiento doble: fragmentaron una imagen usando el esquema (k, n) de SSS y posteriormente insertaron cada fragmento en archivos de audio mediante técnicas de *watermarking*. Aunque no utilizan DCT ni técnicas de verificación de integridad, su enfoque evidencia el potencial del uso de SSS en entornos distribuidos y como mecanismo de confidencialidad robusto. Más directamente relacionado con esta propuesta, el trabajo de Onuma y Miyata [4] combina SSS con inserción en el dominio DCT, aplicando los fragmentos del mensaje secreto en bloques transformados de una imagen portadora. Su método demuestra ser resistente a compresión y permite recuperación del contenido cuando se alcanza el umbral mínimo de fragmentos, lo que valida su idoneidad en escenarios de seguridad visual. A diferencia de su enfoque, la presente propuesta aplica el esquema de Shamir Secret Sharing directamente sobre la imagen ya marcada con la DCT, protegiendo así la totalidad del contenido visual. Además, incorpora un mecanismo de validación de integridad visual mediante hashes perceptuales, lo cual permite verificar que la imagen reconstruida mantiene su estructura visual original, incluso ante posibles transformaciones leves.

Para garantizar la integridad visual de la marca de agua recuperada, técnicas de *perceptual hashing* han demostrado ser altamente eficaces. Estas generan valores hash similares para imágenes visualmente semejantes, permitiendo detectar modificaciones que no necesariamente implican diferencias binarias exactas. Verlekar y Correia [6] emplearon un hash perceptual sobre imágenes biométricas para identificar la dirección de caminata de usuarios, midiendo la similitud con la distancia de Hamming. Su enfoque resultó robusto ante variaciones en la apariencia como ropa o mochilas, y computacionalmente eficiente. Complementariamente, Matej, Grd y Tomicic [1] estudiaron el impacto de distintas transformaciones de imagen sobre valores de hash perceptual, concluyendo que operaciones como compresión o escalado generan pequeñas variaciones, mientras que rotaciones o filtros intensos producen disimilitudes mayores. Estos resultados respaldan el uso de perceptual hashing como herramienta eficaz para verificar marcas de agua visuales robustas.

En conjunto, los trabajos revisados validan la pertinencia de combinar técnicas de esteganografía en dominio DCT, esquemas de compartición segura como SSS, y verificación mediante hash perceptual. Esta integración da lugar a una solución integral que atiende los principales requerimientos de seguridad en imágenes digitales: autenticidad, confidencialidad e integridad visual.

III. Fundamentos Teóricos

Para comprender adecuadamente el sistema propuesto, es necesario introducir los fundamentos matemáticos y criptográficos que lo sustentan. La solución integra tres pilares fundamentales: la Transformada Discreta del Coseno (DCT) para la inserción de marcas de agua, el esquema criptográfico Shamir Secret Sharing (SSS) para fragmentar la imagen de manera segura, y la validación visual mediante hashes perceptuales.

La Transformada Discreta del Coseno (DCT) permite convertir una imagen desde el dominio espacial al dominio de

la frecuencia, facilitando la manipulación de sus componentes espectrales. En este proyecto, se aplica la DCT bidimensional sobre bloques de 8×8 píxeles de una imagen en formato RGB. Esta transformación genera una matriz de coeficientes de frecuencia que describe el contenido visual del bloque en distintos niveles de detalle.

La ecuación general de la DCT bidimensional es:

$$F(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cdot \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cdot \cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{1}$$

Donde:

- $f(x,y)$ representa el valor de intensidad del píxel ubicado en la posición (x,y) dentro de un bloque de tamaño $N \times N$ (en este caso, $N = 8$).
- $F(u,v)$ es el coeficiente DCT correspondiente a la frecuencia espacial (u,v).
- $\alpha(k)$ es un factor de normalización, definido como:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{si } k = 0 \\ \sqrt{\frac{2}{N}} & \text{si } k > 0 \end{cases}$$

- x y y recorren las posiciones horizontales y verticales del bloque, mientras que u y v representan los índices de frecuencia horizontal y vertical, respectivamente.

Para reconstruir el bloque original a partir de sus coeficientes, se utiliza la Transformada Inversa del Coseno (IDCT), definida como:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)F(u,v) \cdot \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cdot \cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{2}$$

En el contexto de este trabajo, el algoritmo de inserción de la marca de agua se basa en la modificación controlada de pares de coeficientes de frecuencia media seleccionados dentro de cada bloque, como por ejemplo (u_1, v_1) y (u_2, v_2). La codificación de un bit se realiza comparando sus valores de magnitud; si el bit a insertar es '1', se asegura que $F(u_1, v_1) > F(u_2, v_2)$, y viceversa. Además, se impone un umbral de robustez x para que la diferencia entre ambos coeficientes sea significativa:

$$|F(u_1, v_1) - F(u_2, v_2)| > x \tag{3}$$

Por otro lado, el esquema *Shamir Secret Sharing (SSS)* es un método criptográfico que permite dividir un dato en múltiples fragmentos denominados *shares*, de modo que solo un subconjunto mínimo de ellos (definido por un umbral k) sea necesario para su reconstrucción. En este trabajo, se aplica SSS a cada píxel de la imagen previamente marcada con la DCT, con el objetivo de garantizar la confidencialidad del contenido durante su distribución. De esta manera, ningún fragmento individual proporciona información útil, y únicamente con al menos k de n acciones es posible recuperar la información original.

Matemáticamente, este esquema se basa en la construcción de un polinomio de grado $k-1$ sobre un campo finito, definido como:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \tag{4}$$

En esta expresión, $f(x)$ es el polinomio secreto generado para representar el valor de un píxel, a_0 corresponde al valor original del píxel (es decir, el secreto a proteger), y los coeficientes $a_1, a_2, \ldots, a_{k-1}$ son seleccionados de manera aleatoria para ocultar el contenido. Cada *share* se obtiene evaluando el polinomio en un punto x_i, obteniendo así un par (x_i, y_i), donde $y_i = f(x_i)$. El conjunto de estos n pares se distribuye como fragmentos seguros de la imagen.

Para recuperar el secreto (el valor original del píxel), se emplea interpolación de Lagrange utilizando al menos k pares distintos. La recuperación del valor en $x = 0$ (es decir, $f(0) = a_0$) se realiza mediante la siguiente expresión:

$$f(0) = \sum_{j=1}^{k} y_j \cdot \prod_{\substack{1 \le m \le k \\ m \neq j}} \frac{-x_m}{x_j - x_m} \mod p \tag{5}$$

Donde (x_j, y_j) representan los pares seleccionados para la reconstrucción y p es un número primo que define el campo finito en el que se llevan a cabo las operaciones. En este proyecto, se utiliza $p = 257$, ya que permite representar valores mayores a 255 sin colisión en imágenes de 8 bits por canal.

Una vez que la imagen marcada ha sido reconstruida a partir de los *shares*, es necesario verificar que su contenido visual no haya sido alterado de manera significativa durante el proceso. Para ello, se utiliza un *hash perceptual*, generado con funciones como `phash()`, que produce una firma binaria de 64 bits que representa las características visuales de la imagen. Esta firma se calcula tanto para la marca de agua original como para la marca recuperada, y se comparan mediante la distancia de Hamming. La distancia de Hamming D_H se define como el número de bits distintos entre dos cadenas binarias del mismo tamaño y se calcula como:

$$D_H(H_1, H_2) = \sum_{i=1}^{n} (H_1[i] \oplus H_2[i]) \tag{6}$$

donde H_1 y H_2 son los hashes correspondientes a la imagen original y la reconstruida respectivamente, $n = 64$ es la longitud del hash, y $\oplus$ indica la operación XOR bit a bit. Un valor bajo de D_H implica alta similitud visual entre ambas imágenes, lo cual es deseable para validar la autenticidad del contenido reconstruido.

La tabla I presenta la interpretación de la distancia de Hamming según su valor. En este trabajo, se establece como umbral aceptable un valor de $D_H \leq 5$, lo cual indica una similitud visual muy alta.

Tabla I
INTERPRETACIÓN DE LA DISTANCIA DE HAMMING EN HASHES PERCEPTUALES DE 64 BITS

Distancia de Hamming	Similitud Visual
0	Idéntica
1 – 5	Muy alta
6 – 15	Alta
16 – 30	Media
31 – 45	Baja
46 – 64	Muy baja

IV. METODOLOGÍA

La metodología propuesta se compone de cuatro fases secuenciales que combinan técnicas de esteganografía, criptografía y verificación de integridad visual para la protección robusta de imágenes digitales. Cada fase contribuye a garantizar al menos uno de los tres servicios fundamentales de seguridad: autenticidad, confidencialidad e integridad.

En la primera fase se ilustra en la figura 1, donde se lleva a cabo la inserción de una marca de agua invisible en la imagen portadora utilizando la Transformada Discreta del Coseno (DCT). Para ello, se emplea una imagen pequeña (por ejemplo, de 16×16 píxeles) como marca visual. La imagen original se divide en bloques de 8×8 píxeles, sobre los cuales se aplica la DCT. En cada bloque, se seleccionan pares de coeficientes de frecuencia media, como $(2, 1)$ y $(3, 3)$, y se insertan bits del mensaje ajustando su relación de magnitud. Finalmente, se aplica la IDCT para reconstruir la imagen con la marca embebida. Esta fase proporciona autenticidad al insertar un identificador oculto de manera robusta e imperceptible.

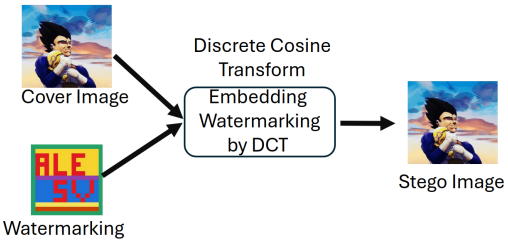


Figura 1. Fase 1: Inserción de la Marca de Agua en DCT

La segunda fase, ilustrada en la figura 2, se enfoca en proteger la confidencialidad de la imagen marcada mediante la técnica criptográfica de Shamir Secret Sharing (SSS). Para ello, la imagen marcada, que originalmente se encuentra representada como una matriz tridimensional de dimensiones $H \times W \times C$ (donde H es la altura, W el ancho y $C = 3$ representa los canales de color RGB), se convierte en un vector unidimensional de longitud $H \cdot W \cdot C$. Cada elemento de este vector corresponde al valor de intensidad de un píxel en un canal específico (rojo, verde o azul).

Una vez vectorizada la imagen, para cada valor de píxel p_i se genera un polinomio aleatorio de grado $k - 1$, donde el término independiente es igual al valor del píxel $a_0 = p_i$, y los coeficientes restantes son seleccionados aleatoriamente. Cada polinomio se evalúa en n puntos distintos $x_1, x_2, ..., x_n$, generando así n imágenes independientes conocidas como *shares*, cada una con la misma dimensión que la imagen original.

Dado que la evaluación del polinomio puede generar valores superiores a 255 (límite de representación en imágenes de 8 bits), se aplica una reducción módulo 257 para ajustar los valores al rango permitido.Este procedimiento asegura que ningún subconjunto de menos de k *shares* pueda revelar información significativa sobre el contenido original, cumpliendo así con los principios de confidencialidad establecidos por el esquema de Shamir.

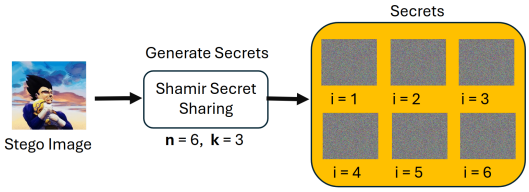


Figura 2. Fase 2: Fragmentación Segura con Shamir Secret Sharing

En la tercera fase que se muestra en la figura 3 se lleva a cabo la reconstrucción de la imagen marcada y la extracción de la marca de agua. A partir de cualquier subconjunto de k acciones, se recupera la imagen original mediante interpolación de Lagrange sobre los valores de píxel codificados. Posteriormente, la imagen reconstruida se divide en bloques, se aplica nuevamente la DCT, y se recuperan los bits insertados en los coeficientes de frecuencia media. Este procedimiento permite recuperar la marca de agua visual y evaluar la autenticidad del contenido.

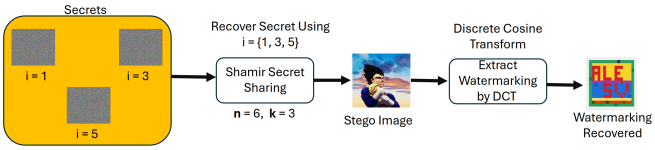


Figura 3. Fase 3: Reconstrucción de la Imagen y Extracción de la Marca de Agua

Por otro lado, la cuarta fase que se observa en la figura 4 consiste en la verificación de la integridad visual de la imagen reconstruida utilizando hashes perceptuales. Se calcula el hash perceptual (`phash`) tanto de la imagen original como de la reconstruida, y se mide su distancia de Hamming. Si dicha distancia se encuentra por debajo de un umbral (por ejemplo, 10), se concluye que la imagen no ha sido modificada visualmente de forma significativa. Esta etapa permite validar que la marca de agua extraída es coherente con la imagen original y actúa como mecanismo final de verificación de integridad.

En conjunto, las técnicas aplicadas en cada fase aportan beneficios complementarios que fortalecen la seguridad del sistema propuesto. La inserción de la marca de agua en el dominio DCT garantiza autenticidad mediante un identificador robusto y oculto; la fragmentación criptográfica basada en Shamir Secret Sharing (SSS) refuerza la confidencialidad al impedir el acceso a la imagen sin el umbral mínimo de fragmentos; y la verificación mediante hash perceptual

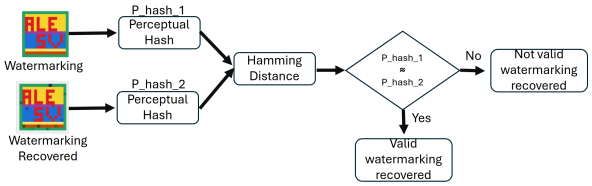


Figura 4. Fase 4: Verificación de la Marca de Agua con Hash Perceptual

proporciona un mecanismo efectivo de validación estructural para preservar la integridad visual. Esta arquitectura integral responde a necesidades críticas en entornos donde la confianza en el contenido digital es esencial, y sienta las bases para evaluar el desempeño del sistema en condiciones prácticas, como se detalla en la siguiente sección.

V. Resultados y Discusión

Esta sección presenta los resultados obtenidos tras la implementación del sistema propuesto, evaluando su eficacia desde tres enfoques: imperceptibilidad, confidencialidad y verificación de integridad visual.

V-A. Inserción de la Marca de Agua con DCT

En la figura 5 se comparan la imagen original 5a, la imagen con marca de agua insertada 5b y la marca utilizada 5c. A simple vista, la imagen modificada conserva su apariencia, indicando una inserción imperceptible.

(a) Imagen portadora (cover) (b) Imagen stego (c) Marca de agua incrustada

Figura 5. De izquierda a derecha: imagen original, imagen con marca de agua (stego) y marca utilizada.

Para cuantificar la distorsión visual, se calcularon los niveles de *PSNR* por canal, los cuales se presentan en la tabla II.

Canal de Color	PSNR (dB)
Azul (Blue)	38.78
Verde (Green)	37.89
Rojo (Red)	37.89

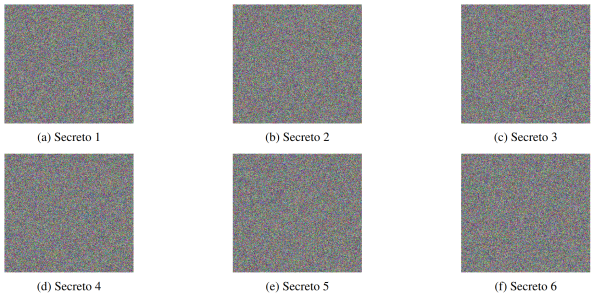
Tabla II
Resultados de PSNR entre la imagen original y la imagen con marca de agua (stego)

Valores superiores a 37 dB reflejan alta fidelidad, lo cual confirma que la inserción en frecuencias medias con DCT es efectiva para garantizar la imperceptibilidad.

V-B. Fragmentación Segura con Shamir Secret Sharing

Se aplicó el esquema con parámetros $n = 6$, $k = 3$. En la figura 6 se muestran las acciones generadas (shares), las cuales no revelan información visual.

Este resultado confirma la confidencialidad: ningún share individual permite recuperar la imagen original.

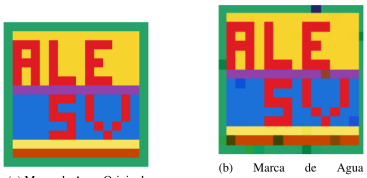


(a) Secreto 1 (b) Secreto 2 (c) Secreto 3
(d) Secreto 4 (e) Secreto 5 (f) Secreto 6

Figura 6. Acciones generadas mediante Shamir Secret Sharing, con $n = 6$ y $k = 3$.

V-C. Reconstrucción y Extracción de la Marca de Agua

Con tres acciones seleccionadas aleatoriamente $k = \{1, 3, 5\}$ de $n = 6$ se reconstruyó y extrajo exitosamente la marca de agua, tal como se muestra en figura 7.



(a) Marca de Agua Original (b) Marca de Agua Recuperada

Figura 7. Marca original (izquierda) y recuperada (derecha) tras la reconstrucción.

El nivel de distorsión se evaluó con PSNR y diferencia píxel a píxel:

Canal	PSNR (dB)
Azul	28.49
Verde	28.88
Rojo	25.90

Tabla III
PSNR entre marcas original y recuperada

Métrica	Valor
Media	1.5365
Máxima	132
Mínima	0
Desviación Est.	10.5881

Tabla IV
Diferencia píxel a píxel

Las tablas III y IV evidencian que la marca fue recuperada con calidad aceptable, pese al ruido generado por interpolación y cuantización en los algoritmos de inserción y SSS.

V-D. Verificación de Integridad con Hash Perceptual

Se utilizó `phash()` para comparar la marca original y la recuperada. Los resultados se muestran en la tabla V.

La baja distancia de *Hamming* 4 de 64 bits valida la autenticidad visual de la imagen recuperada, cumpliendo con el umbral establecido ≤ 5, tal como se menciono en la tabla I.

En resumen, los resultados obtenidos evidencian la efectividad de la metodología propuesta, garantizando autenticidad, confidencialidad e integridad visual de imágenes digitales.

Hash 1	8585c47a7bf13296
Hash 2	8585c47a6be13e96
Distancia Hamming	4
Similitud (%)	94 %
Interpretación	Visualmente muy similares

Tabla V
VERIFICACIÓN MEDIANTE HASH PERCEPTUAL

VI. CONCLUSIÓN

Este trabajo propone un esquema integral para la protección de imágenes digitales, combinando esteganografía en el dominio DCT, compartición segura mediante *Shamir Secret Sharing* (SSS) y verificación visual basada en *hashes* perceptuales. Cada una de estas fases contribuye a garantizar servicios fundamentales de seguridad:

- Autenticidad: proporcionada por la marca de agua embebida y validada posteriormente mediante comparación con su versión recuperada.
- Confidencialidad: asegurada por la fragmentación criptográfica del contenido, que impide la reconstrucción del secreto con menos de k acciones.
- Integridad Visual: validada por el análisis de similitud utilizando funciones hash perceptuales y métricas como PSNR y diferencia píxel a píxel.

Los resultados experimentales obtenidos demuestran que el sistema propuesto es capaz de preservar la calidad visual de la imagen, proteger el acceso al contenido y detectar modificaciones no autorizadas. Los valores de PSNR superiores a 37 dB entre la imagen original y la marcada, la distancia de Hamming igual a 4 entre los hashes de la marca de agua original y la recuperada, así como la diferencia promedio de píxeles menor a 2, validan la efectividad del enfoque desarrollado.

En conjunto, este sistema representa un aporte relevante en el diseño de mecanismos de seguridad para imágenes, con potencial aplicación en contextos sensibles como medicina, videovigilancia o protección de propiedad intelectual.

VII. TRABAJO FUTURO

El presente trabajo establece una base sólida para la protección de imágenes digitales mediante técnicas de esteganografía, compartición segura y verificación visual. Como líneas futuras, se propone optimizar el algoritmo de inserción en el dominio DCT para incrementar la capacidad sin comprometer la calidad visual, así como incorporar códigos de corrección de errores que permitan una recuperación más fiel de la marca de agua. Asimismo, se sugiere ampliar las pruebas experimentales para evaluar de forma sistemática la robustez del esquema frente a ataques intencionales y comunes en watermarking, tales como compresión con pérdida (por ejemplo, JPEG), adición de ruido (como ruido gaussiano o sal y pimienta), filtrado, recorte y rotación. Este análisis permitirá determinar con mayor precisión los límites de resistencia del sistema propuesto en escenarios reales de manipulación de imágenes. Además, se plantea explorar variantes avanzadas de *Shamir Secret Sharing* para control de acceso jerárquico o entornos distribuidos. Por otro lado, para robustecer la etapa de verificación de integridad visual, se considera de interés comparar distintos algoritmos de hashing perceptual (como aHash y dHash), así como incorporar enfoques de deep hashing basados en redes neuronales, los cuales podrían ofrecer mayor resiliencia frente a manipulaciones complejas o degradaciones del contenido visual.

Finalmente, se plantea la extensión del presente esquema hacia otros dominios multimedia, como video o audio, donde garantizar autenticidad e integridad visual es fundamental para aplicaciones críticas.

VIII. AGRADECIMIENTOS

Agradecemos a SECIHTI por el apoyo financiero brindado mediante becas de posgrado, al INAOE por proporcionar un entorno académico de excelencia, y al cuerpo docente del programa de Maestría en Ciencias y Tecnologías de Seguridad por su valiosa formación. Reconocemos especialmente a la profesora responsable de la asignatura de Seguridad Avanzada y Marcas de Agua por su orientación, así como a los autores y desarrolladores de recursos técnicos utilizados en este proyecto.

REFERENCIAS

[1] Ferenčak, Matej, Petra Grd y Igor Tomičić: *The Impact of Image Processing on Perceptual Hash Values*. En *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, 2023.

[2] Katzenbeisser, Stefan y Fabien A. Petitcolas: *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Inc., Norwood, MA, USA, 1st edición, 2000.

[3] Kaçamak, Salajdin y Arban Uka: *Sound steganography using Shamir secret sharing scheme*. En *2017 6th Mediterranean Conference on Embedded Computing (MECO)*, 2017.

[4] Onuma, K. y S. Miyata: *A Proposal for Correlation-based Steganography Using Shamir's Secret Sharing Scheme and DCT Domain*. International Journal of Engineering and Advanced Technology (IJEAT), 9(1), 2019.

[5] Srivastava, L. y Garg H.: *Security of Image Using Watermarking Techniques and Visual Cryptography*. En *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*. IEEE, 2023.

[6] Tanmay, T. Verlekar y Paulo L. Correia: *Walking Direction Identification Using Perceptual Hashing*. En *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015.

Ejecución Segura y Verificación de Firmware Cifrado en Tiempo de Ejecución en Entornos IoT

Alejandro Salinas Victorino
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
alejandro.salinasv@inaoe.mx

Raziel César Campos Sánchez
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
rcampos@inaoe.mx

José de Jesús Zapata Lara
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
josedj.zapata@inaoe.mx

***Resumen*—En entornos del Internet de las Cosas (IoT), los dispositivos embebidos basados en microcontrolador como el ESP32 suelen carecer de mecanismos robustos de protección del firmware, exponiéndolos a ataques de clonación, ingeniería inversa y robo de identidad. Este trabajo presenta una arquitectura de ejecución segura de firmware en tiempo de ejecución (run time) basada en MicroPython, donde los módulos funcionales del firmware se almacenan cifrados en memoria flash y son validados mediante Hash Message Authentication Code (HMAC) antes de su ejecución dinámica. La solución garantiza que solo firmware autorizado puede ejecutarse, mitigando así riesgos de modificación o sustitución maliciosa del código. Se desarrolla un ecosistema modular que integra sensores, actuadores y comunicación HTTP, y se valida la propuesta mediante análisis de memoria flash en escenarios comparativos. Los resultados muestran que, a diferencia de una implementación convencional de cargar el firmware en texto claro al dispositivo IoT, la arquitectura propuesta previene la exposición de datos sensibles y refuerza la integridad y autenticidad del firmware, incluso ante ataques con acceso físico al dispositivo.**

***Palabras clave*—Seguridad IoT, ESP32, MicroPython, ejecución en run time, HMAC, firmware cifrado, autenticación de firmware, memoria flash, ingeniería inversa, robo de identidad.**

I. Introducción

En el ámbito del Internet de las Cosas (IoT), proteger el firmware de los dispositivos conectados es una prioridad clave para salvaguardar la seguridad del ecosistema. Dispositivos embebidos como el ESP32 son ampliamente utilizados por su bajo costo, eficiencia energética y conectividad inalámbrica, pero carecen de mecanismos robustos de protección, lo que los hace vulnerables a clonación, ingeniería inversa e inyección de código malicioso. Si bien existen mecanismos como *Secure Boot* y firmas digitales para mitigar estos riesgos, su implementación suele requerir hardware especializado o configuraciones complejas, lo que los hace poco viables en dispositivos de bajo costo o en fases tempranas de desarrollo.

Ante esto, se propone una arquitectura segura de ejecución de firmware en tiempo de ejecución sobre ESP32 usando MicroPython. Esta solución almacena el firmware cifrado en la memoria flash y utiliza códigos HMAC para verificar su integridad y autenticidad antes de su ejecución mediante `exec()`. Solo si la verificación es exitosa, el código se descifra y ejecuta dinámicamente, garantizando que solo firmware autorizado sea cargado.

La propuesta se valida en un sistema modular con sensores y actuadores, comparando un escenario de firmware en texto claro con otro que implementa la arquitectura propuesta. Los resultados muestran que se mitigan ataques de extracción de memoria flash y se protege tanto la lógica del firmware como los datos sensibles, reduciendo riesgos como el robo de identidad. Además, el trabajo se alinea con la guía *NIST-SP-800-193* del NIST, al implementar cifrado de datos críticos y verificación de integridad para mantener los dispositivos en un estado seguro frente a ataques.

El documento se organiza así: la Sección II revisa trabajos relacionados; la Sección III expone los conceptos clave; la Sección IV detalla la metodología de implementación; la Sección V presenta los resultados experimentales; la Sección VI ofrece la conclusión; y la Sección VII plantea futuras mejoras.

II. Trabajo relacionado

La ejecución segura de firmware en dispositivos embebidos ha sido ampliamente abordada desde distintas perspectivas, principalmente mediante mecanismos de *secure boot*, firmas digitales y cifrado. Sin embargo, muchos de estos enfoques dependen de hardware especializado o suponen un entorno de producción controlado, lo cual no siempre es viable en dispositivos de bajo costo o en escenarios dinámicos como el IoT.

Molcut et al. [3], destacan la necesidad de garantizar que los sistemas embebidos ejecuten únicamente firmware autorizado. El trabajo revisa distintas estrategias de protección como el uso de funciones hash, cifrado simétrico/asimétrico y dispositivos criptográficos dedicados. Asimismo, enfatiza que la combinación de mecanismos como verificación de integridad, autenticación y cifrado es esencial para evitar ataques como la sobrescritura de memoria o el uso de firmware malicioso.

Por otro lado, *Wang and Yan* [4] clasifican los principales esquemas de arranque seguro, incluyendo enfoques basados

en hardware (como TPM o TrustZone) y software (como verificación de firmas digitales). Aunque eficaces, estos métodos suelen implicar costos adicionales o altos requerimientos computacionales, lo que limita su adopción en dispositivos con recursos restringidos. En contraste, la solución propuesta en este trabajo busca ofrecer un mecanismo liviano, ejecutado en tiempo de ejecución mediante *MicroPython*, que permite validar y descifrar firmware de forma dinámica usando HMAC.

Younis et al. [5] presentan un enfoque basado en hardware para la validación de la integridad del firmware que garantice el arranque seguro para sistemas IoT con recursos computacionales limitados y para los cuales la variable del costo es importante. La solución propuesta incorpora el uso de PUF's para la identificación de forma única de cada dispositivo IoT y de esta forma establecer una raíz de confianza. Los resultados obtenidos demostraron que la solución propuesta es resistente ante ataques "man-in-the-middle", suplantación de identidad y reproducción de mensajes y en términos de desempeño presenta una carga mínima de recursos de hardware, consumo de energía y latencia.

Feng et al. [1] presentan un esquema de arranque para dispositivos IoT que utiliza Criptografía Basada en Identidad (IBC), el cual ha sido implementado mediante algoritmos criptográficos de curva elíptica. El trabajo propuesto utiliza autenticación bidireccional sin certificado mediante una entidad central que genera y distribuye las claves privadas para los dispositivos y para detectar la manipulación de dispositivos se utiliza la tecnología de PUF's. El esquema propuesto esta integrado por las fases de inicialización del sistema, generación de la imagen de arranque y la validación e inicio del dispositivo.

Finalmente, *Kumar et al.* [2] proponen un marco integral que incluye múltiples capas de autenticación, autorización y cifrado para proteger el proceso de actualización de firmware. Aunque su enfoque está más orientado a escenarios industriales y de infraestructura crítica, comparte la motivación central de asegurar que solo firmware legítimo y validado sea instalado y ejecutado. En ese sentido, la solución aquí presentada se posiciona como una alternativa modular y adaptable para entornos de bajo consumo, reforzando la seguridad desde la capa de ejecución.

III. Preliminares

Esta sección presenta los conceptos esenciales que sustentan la propuesta de ejecución segura de firmware autorizado en dispositivos IoT basados en ESP32 y ejecutados con MicroPython.

- Ejecución en tiempo de ejecución (Runtime Execution): Consiste en interpretar y ejecutar instrucciones dinámicamente durante la operación del sistema, lo cual permite actualizaciones flexibles, carga bajo demanda y ejecución modular del firmware, características especialmente útiles en entornos IoT.
- HMAC para autorización: El Hash Message Authentication Code (HMAC), basado en funciones hash como SHA-256 y una clave secreta, garantiza la integridad y autenticidad del firmware. Solo los módulos con HMAC válido son descifrados y ejecutados.
- Almacenamiento seguro: Hace referencia a la protección de datos sensibles, como claves y configuraciones, en zonas cifradas de la memoria flash del ESP32. Esto evita su exposición ante accesos físicos no autorizados.
- Uso de `exec()` **en MicroPython:** Esta función permite ejecutar código Python desde texto en tiempo de ejecución. En esta propuesta, se emplea para descifrar y ejecutar dinámicamente los módulos del firmware, tras verificar su HMAC, manteniendo así una arquitectura segura y flexible.
- Riesgos de clonación y exposición de información crítica: en entornos IoT, donde los dispositivos suelen estar físicamente expuestos, es común enfrentar ataques como la clonación de firmware y la extracción de datos desde la memoria flash. Un adversario puede replicar el firmware embebido mediante lectura directa o ingeniería inversa, comprometiendo tanto la propiedad intelectual como la integridad del sistema. Asimismo, almacenar claves o configuraciones sensibles sin protección adecuada expone al dispositivo a técnicas como *memory dumping* o *glitching*, vulnerando los principios de confidencialidad, integridad y autenticación.

IV. Metodología

A continuación, se describe la estrategia de diseño e implementación de la arquitectura propuesta para la ejecución segura de firmware autorizado en dispositivos IoT basados en ESP32 utilizando MicroPython. La metodología considera la estructura modular del firmware, el uso de cifrado simétrico y autenticación mediante HMAC, así como la ejecución dinámica del código en tiempo de ejecución.

IV-A. Descripción del firmware

El firmware desarrollado implementa una arquitectura modular orientada a objetos, donde las funcionalidades están encapsuladas en archivos independientes organizados por clases. Esta estructura permite escalabilidad, reutilización y ejecución dinámica de componentes mediante `exec()`. Los módulos principales incluyen: comunicación (`WifiControl.py`, `http_communication.py`), autenticación (`Authentication3.py`), sensores ambientales (`temperature_sensor.py`, `microphone_sensor.py`), procesamiento de datos (`Processing_data.py`), actuadores (`led_semaphore.py`, `IR_send.py`), integración funcional (`Device.py`) y orquestación general (`main.py`).

Adicionalmente se incluye un archibo de configuración `Config.py`, que centraliza parámetros sensibles como pines, llaves de sesión, credenciales Wi-Fi y claves criptográficas AES para las comunicaciones. Esta organización permite mantener los módulos cifrados hasta su ejecución, fortaleciendo la seguridad del sistema.

IV-B. Ejecución segura del firmware en tiempo de ejecución

Se diseñó una arquitectura que permite almacenar y ejecutar el firmware de forma segura. Todos los módulos (excepto `main.py` y `Run_Time_Encryption_Hmac.py`) se encuentran cifrados y verificados mediante HMAC, figura 1. Durante la ejecución:

1. El archivo cifrado es cargado desde memoria flash.
2. Se descifra en RAM usando AES-CBC.
3. Se verifica su integridad con HMAC-SHA256.
4. Si la verificación es exitosa, se ejecuta dinámicamente con `exec()`.

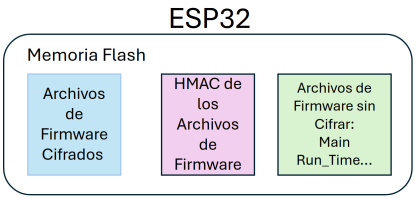


Figura 1. Archivos almacenados en la memoria flash del ESP32.

Este proceso asegura que el firmware nunca reside en texto claro en memoria no volátil, mitigando ataques como clonación o lectura directa de claves criptográficas.

Cabe mencionar que el uso de la función exec() en MicroPython, si bien ofrece flexibilidad para ejecutar código dinámico en tiempo de ejecución, también representa un riesgo significativo si se emplea sin mecanismos de control, ya que permite la ejecución arbitraria de cualquier cadena de texto interpretada como código. Esto podría ser explotado por un atacante para insertar y ejecutar código malicioso, especialmente si el contenido proviene de fuentes no confiables o ha sido modificado en la memoria del dispositivo. Para mitigar este riesgo, la solución propuesta implementa un sistema de validación basado en HMAC (Hash-based Message Authentication Code) que verifica la integridad y autenticidad de cada módulo antes de su ejecución. Solo aquellos fragmentos de firmware cuyo HMAC coincida con el esperado son descifrados y ejecutados mediante exec(), lo que garantiza que únicamente código autorizado y no alterado pueda formar parte del sistema activo, reduciendo significativamente la superficie de ataque y mejorando la seguridad en entornos IoT expuestos.

IV-B1. Cifrado y verificación del firmware:

- Cifrado: Cada archivo en texto claro (por ejemplo, `Device.txt`) es cifrado con AES-CBC y codificado en base64 (`enc_Device.txt`), figura 2.
- HMAC: Se genera un HMAC-SHA256 sobre el archivo original, almacenado como `hmac_Device.txt`, figura 2.
- Verificación: En tiempo de ejecución, se descifran los archivos y el HMAC recibido se compara con el generado localmente. Solo si coinciden, el código es ejecutado, figura 3.

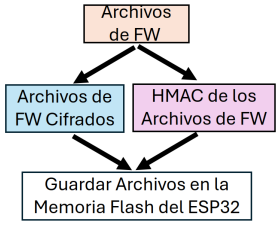


Figura 2. Cifrado de archivos y generación de HMAC del Firmware.

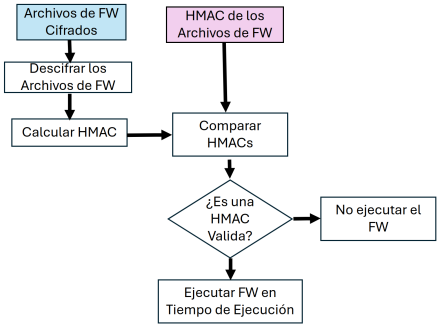


Figura 3. Descifrado y verificación del Firmware para su ejecución.

IV-B2. Componentes en texto claro: Solo dos scripts residen en texto claro en el sistema:

- `Run_Time_Encryption_Hmac.py`: contiene las funciones de cifrado, descifrado y verificación de HMAC.
- `main.py`: orquesta la carga, validación y ejecución de los módulos dinámicos.

En resumen, este enfoque modular garantiza la ejecución exclusiva de firmware autorizado, mitigando riesgos de modificación, ingeniería inversa y clonación, y demostrando su viabilidad en entornos IoT con microcontroladores de recursos limitados como el ESP32.

IV-C. Diferencias Con Secure Boot en ESP32

A diferencia del mecanismo de Secure Boot del ESP32, que requiere configuración en fase de fabricación y permanece estático durante el ciclo de vida del dispositivo, la solución propuesta ofrece una alternativa flexible y adaptable en tiempo de ejecución. Si bien Secure Boot garantiza la integridad del firmware desde el arranque, nuestra arquitectura permite validar y ejecutar módulos dinámicamente, lo que resulta especialmente útil en prototipos o entornos IoT donde el firmware cambia con frecuencia y no se dispone de acceso a eFuses o herramientas de producción avanzadas

V. Resultados y Discusión

En esta sección se presentan los resultados obtenidos a partir de una evaluación comparativa entre dos escenarios implementados sobre el microcontrolador ESP32. El objetivo principal es demostrar la efectividad de la solución propuesta para proteger el firmware y los datos sensibles almacenados

en el dispositivo frente a ataques de extracción directa desde la memoria flash.

- **Escenario 1:** Ejecución del firmware en texto claro, sin mecanismos de cifrado ni verificación de integridad mediante HMAC. En este caso, todos los archivos del sistema (incluyendo configuraciones y credenciales) se almacenan directamente en la memoria flash del ESP32.
- **Escenario 2:** Aplicación de la arquitectura propuesta, en la cual el firmware y los archivos de configuración se almacenan en formato cifrado dentro de la memoria flash, y son verificados y ejecutados dinámicamente en tiempo de ejecución (*run time*) mediante MicroPython.

V-A. Extracción de Memoria Flash y Exposición de Datos Sensibles

Para evidenciar las vulnerabilidades asociadas al almacenamiento en texto claro, se realizó una lectura (dump) completa de la memoria flash en ambos escenarios utilizando la herramienta `esptool.py`, seguida de un análisis con la utilidad `strings` de Linux para la extracción de texto legible. Esta metodología simula un ataque físico orientado a clonar el dispositivo o extraer parámetros críticos de configuración.

```
Password = "password123"
WifiPassword = 'Alex1234'
AES_Password = "Alex1234"
```

Figura 4. Contraseña localizada mediante análisis de memoria flash en el Escenario 1.

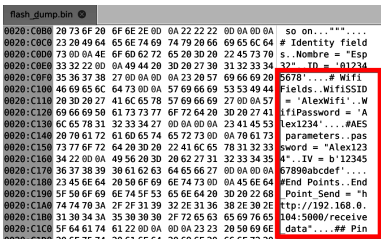


Figura 5. Fragmento de la captura de memoria flash en el Escenario 1, mostrando configuraciones en texto claro.

Como se observa en las figuras 4 y 5, el análisis de la memoria en el escenario 1 revela múltiples elementos sensibles almacenados en texto claro. Entre ellos se encuentran credenciales como contraseñas de red y archivos de configuración completos. Este tipo de exposición representa una amenaza crítica de seguridad, ya que facilita ataques de clonación de firmware, suplantación de identidad de dispositivos e ingeniería inversa del comportamiento del sistema.

Además, el uso de patrones comunes como `Password`, `SSID` o `Token` permite automatizar la extracción de secretos mediante listas predefinidas de búsqueda, lo que aumenta aún más el riesgo de compromiso.

En contraste, la figura 6 presenta los resultados del análisis realizado sobre la memoria flash en el escenario 2. En este caso, la búsqueda del término `Password` no arroja resultados

```
Wifi Invalid Password
Passwords do not match
Password:
```

Figura 6. Resultado del análisis de memoria flash en el Escenario 2: sin exposición de contraseñas.

visibles, lo que confirma que los archivos sensibles han sido correctamente cifrados y que su contenido no es accesible directamente desde la memoria.

Este resultado valida la efectividad de la arquitectura propuesta, la cual mitiga significativamente los riesgos asociados a ataques por lectura directa de la memoria flash, al evitar que secretos criptográficos, credenciales o configuraciones sean expuestos en texto claro.

V-B. Ejecución Segura de Firmware mediante Verificación de HMAC

Además de almacenar el firmware en formato cifrado dentro de la memoria flash del ESP32, el dispositivo IoT implementa un mecanismo de autenticación basado en códigos HMAC (Hash-based Message Authentication Code). Antes de ejecutar cualquier módulo durante el tiempo de ejecución, se realiza una verificación del HMAC correspondiente. Esta validación garantiza que el archivo no ha sido alterado y que proviene de una fuente legítima.

```
This is the run time encryption script.
Main process
the class_name is: ConfigParameters
read file hmac file
HMAC mismatch: File integrity check failed!
```

Figura 7. Resultado de la verificación HMAC de un módulo comprometido.

La figura 7 muestra un ejemplo en el cual la verificación HMAC no es satisfactoria. Este resultado puede deberse a una modificación no autorizada del archivo, corrupción en el proceso de transferencia, o la sustitución maliciosa del módulo. En tales casos, el sistema impide la carga del módulo, evitando su ejecución.

Gracias a la estructura modular del firmware, cada archivo es evaluado de manera independiente. Si alguno de los módulos no pasa la verificación HMAC, el sistema continúa operando con el resto de los componentes válidos, descartando únicamente el archivo comprometido. Esta estrategia permite mantener la operación parcial del dispositivo sin comprometer la seguridad del entorno de ejecución.

De este modo, se asegura que únicamente los módulos que han sido previamente autorizados y cuya integridad ha sido confirmada sean ejecutados. Este proceso actúa como un filtro de seguridad en tiempo de ejecución, garantizando que solo firmware autorizado y verificado forme parte del sistema activo del dispositivo IoT.

V-C. Comparativa de Tiempo de Ejecución y Consumo de Memoria RAM

En esta subsección se presenta una comparativa entre la arquitectura propuesta que incorpora la ejecución y verifi-

cación del firmware en tiempo de ejecución frente a una implementación tradicional, en la que el firmware se ejecuta directamente desde memoria flash sin mecanismos de cifrado ni validación de integridad. Los resultados se obtuvieron mediante pruebas sobre un ecosistema IoT real, diseñado para medir parámetros ambientales como temperatura, humedad y nivel de ruido en un datacenter.

Tabla I
DESEMPEÑO DEL DISPOSITIVO IOT

Arquitectura Propuesta	Tiempo de Sensado	Tiempo de Envío - Recepción de Datos	Uso en Memoria RAM
Sí	3.0 s	10.0 s	98.9 KB
No	3.0 s	10.0 s	46.5 KB

Los parámetros considerados para esta evaluación son:

- Tiempo de sensado: Corresponde al intervalo requerido por el ESP32 para adquirir datos desde sensores ambientales conectados.
- Tiempo de envío y recepción de datos: Tiempo total que tarda en transmitir los datos recolectados hacia un servidor remoto, en este caso desde Zapopan, Jalisco, hasta Ciudad Victoria, Tamaulipas.
- Uso de memoria RAM: Cantidad total de memoria utilizada durante la ejecución completa del firmware bajo cada enfoque.

Como se observa en la tabla I, los tiempos de sensado y de comunicación permanecen constantes en ambos enfoques, lo que demuestra que la solución propuesta no introduce latencia perceptible en la operación del dispositivo. Sin embargo, el consumo de memoria RAM se incrementa significativamente más del doble cuando se ejecuta el firmware de manera cifrada y dinámica. Este aumento es atribuible a las operaciones adicionales necesarias para el descifrado, verificación HMAC y ejecución mediante `exec()` en tiempo de ejecución.

A pesar del mayor uso de memoria, este enfoque proporciona una ventaja considerable en términos de seguridad: impide que el firmware se almacene en texto claro en memoria flash, reduciendo significativamente la superficie de ataque ante intentos de clonación, ingeniería inversa o robo de identidad del dispositivo. Por tanto, el incremento en el uso de recursos es un compromiso razonable a cambio de un entorno de ejecución más robusto y resiliente en aplicaciones IoT expuestas a amenazas físicas.

V-D. Riesgos de ingeniería inversa

Uno de los principales objetivos de la ingeniería inversa es analizar un producto, sistema o componente para comprender su funcionamiento interno, conocer su estructura y, eventualmente, replicarlo o modificarlo. Cuando el firmware de un dispositivo embebido como el ESP32 se encuentra almacenado en texto claro, se expone directamente a este tipo de análisis, lo que representa un riesgo significativo para la confidencialidad del diseño y la propiedad intelectual.

La figura 8 ilustra cómo una porción del firmware almacenado en texto claro puede ser recuperada directamente desde

```
        decipher_text = aes_decrypt.decrypt(cipher_text)
        decipher_text = self.unpad(decipher_text)
        return decipher_text
IoT 2
Authors:
    -- Raziel Campos
    -- Jose Zapata
    -- Alejandro Salinas
from Device import DeviceIoT
# TODO: Move the http class to device, only to test
from http_communication import HttpCommunication
import Config
import json
import hashlib
import time
from Authentication3 import Authentication3
from Processing_data import ProcessingData
def main():
    # The object initialization is once per run
```

Figura 8. Firmware en texto claro recuperado desde la memoria flash.

```
NugFthK3KAoVPV6EfOLyE4RdPY+MPmszgo08CevJyP9ubUBwSUsQqtIQzWMf9fa//Sp
lA6nBmNMgm038xYsDyqVB84b4OLbKkgnnlhwLeRHqz1N3t9bJlBW7JamOYvKDL/oRpe
HihhC9q/uEmEPyR+D5to4uiW1RtGJsniZls8m1UEgeS4ht75JQIUaM8xEbAuGtayIEJ
0nAB38ZhL1rDi2/tURg5mGe3lzp0AgmZ5L7bcI+q74ucT3GcFmGai9jyXmRc9ZG8YZb
ykcnZNxvgfzHu5AlltXwKQtJq0K5RM0cC54Hxt1U9S9uxewFnPc4gJYzSE1VuKpQofi
NYwjx51fkIS0s+1KxSaVKzc+uxo6xYLjfMwMz4MQDLNLCjY+VKC7a2le3Vfawsbv97o
yt5oeMGMIuUUCqbLuqqDu1lwTRNBiCs43KVzRtMtVgrUPJPLIRHd570Fb3LnIeDzxzB
Wq/6r0/y2WUfPWTt3HUUTRUMA61bLVJ+A8l+QYDFzDrAHdgI+gh1C8PcH4Huq2PVJKM
4JkmVofUWCz9xBkwsFPmaANU0hZMQqt+EHDmUDjAD/ciNhqPvjyYBjzoLlWPwty+r6b
YmHyYJMKoZBHNAr9MAT8CGk/APS0crr5RcxaiI/e6SShte4NLlVa2DYDANehqgdSeDP
/Y3N/2bh5koocbxKfjkX026J37vCR03aRO9u9BzzncDS09coN6C7/isb0fi4iKsdH6u
NfWVpxRW4GBRzC6P9lBTUH+U/G6bEBpmb0U4tdk3UToaXNZDoi0rXT+PIL29eOLOB9g
uDZj06RKg/TQogVKU8sDQWrcQ1osGVa4a64kB2/CInKpjtpB8A7pM5kgE5+nuayTjw3
yZdd+Zvo83zzH/rx2VL5Cp6r1U7qdOGKoeE5SOlY3kFROM+ozgAqq8UM2hyKkT2rTvp
+o3uga0pD9lPess6UW/w6TLM/y8b0i1VqUn/yXBkHEsRdSURPCbvPIbtvN8CtpgK802
IC+O8J22mUygtbaK1YPVjENr4vqVDfPLYXGCXWiS+Chueb/eGEN7VPee6OxTPiqlCEq
W97gVMuFqWvGJZNVptRZNNpQglS3AtVd+avA9H8UYBKKH516rmAImPBFYUSTcPTTnGX
bIwL6RXB92aXgtCySRVfh+MNaY8aGbsmIvBQ9Us=
This is the main script to run the iot project.
Authors:
    -- Raziel Campos
    -- Jose Zapata
    -- Alejandro Salinas
```

Figura 9. Firmware cifrado recuperado en la lectura de la memoria flash.

la memoria flash del dispositivo. En este caso, se identifican fragmentos correspondientes a los módulos de comunicación y sensores, lo cual permite al atacante inferir el comportamiento del sistema y reconstruir su lógica funcional. Este tipo de exposición facilita la ingeniería inversa del firmware y puede derivar en la clonación del dispositivo, modificación del comportamiento o robo de propiedad intelectual.

Por el contrario, la figura 9 muestra el resultado obtenido al aplicar la solución propuesta. En este escenario, únicamente los archivos no cifrados —como el `main.py` y el módulo de descifrado y verificación— son legibles en la memoria flash. El resto del firmware permanece cifrado, lo que impide su interpretación directa incluso si se accede físicamente al contenido de la memoria. Esta medida representa una defensa efectiva contra técnicas de ingeniería inversa basadas en extracción y análisis de firmware.

En resumen, los resultados demuestran que la solución basada en cifrado simétrico y verificación HMAC en tiempo de ejecución incrementa de manera sustancial la protección del dispositivo frente a ataques físicos. Esta capa adicional de seguridad es especialmente relevante en entornos IoT, donde los dispositivos suelen estar desplegados en ubicaciones no controladas y expuestos a actores maliciosos con acceso físico. Es importante mencionar que el proyecto OWASP IoT Top 10 establece que el almacenamiento no seguro y las configuraciones por default representan riesgos de seguridad críticos para los dispositivos IoT y que la base de conocimiento MITRE EMB3D identifica a la falta de protección y verificación en la ejecución del firmware como una de las

principales amenazas a la seguridad de la capa de software de un sistema embebido. Con base en lo anterior, se plantea que el trabajo desarrollado puede ser utilizado como un elemento funcional para el cumplimiento de las directrices que establece la norma ISA/IEC 62443-4-2 sobre los requerimientos técnicos para fortalecer la seguridad de los sistemas industriales para automatización y control, en específico para salvaguardar la confidencialidad de los datos en los dispositivos y la integridad del sistema.

VI. Conclusión

Este trabajo propuso e implementó una arquitectura de ejecución segura de firmware en tiempo de ejecución, orientada a microcontroladores ESP32 que operan bajo MicroPython, sin depender de mecanismos avanzados como *Secure Boot* o módulos criptográficos embebidos. La solución se basa en el cifrado simétrico AES y en la verificación de HMAC, lo que permite almacenar módulos de firmware cifrados y validar su integridad y autenticidad antes de ejecutarlos dinámicamente mediante `exec()` en Micropython.

Este enfoque garantiza que únicamente firmware autorizado puede ser cargado en el sistema, impidiendo la ejecución de código modificado o malicioso. Además, gracias a su estructura modular, se facilita el descarte selectivo de módulos no válidos sin comprometer el funcionamiento global del dispositivo.

Las pruebas demostraron que almacenar el firmware en texto claro expone gravemente a los dispositivos IoT, permitiendo la extracción de contraseñas, llaves criptográficas y configuraciones sensibles desde la memoria flash. Este tipo de vulnerabilidades abre la puerta a ataques de clonación, robo de identidad y suplantación de dispositivos.

En el contexto de la Industria 5.0 —donde la confiabilidad, conectividad y resiliencia son pilares fundamentales— esta arquitectura se posiciona como una solución eficaz para proteger sensores inteligentes, robots colaborativos, sistemas de mantenimiento predictivo y equipos industriales críticos frente a amenazas físicas o lógicas. Su diseño ligero, adaptable y de bajo costo ofrece un camino viable para fortalecer la ciberseguridad en dispositivos embebidos, extendiendo su utilidad más allá del ámbito académico hacia aplicaciones reales en entornos industriales inteligentes.

VII. Trabajo futuro

Este trabajo ha demostrado la viabilidad de ejecutar firmware seguro en dispositivos IoT mediante cifrado simétrico y verificación HMAC en tiempo de ejecución. Sin embargo, existen oportunidades claras de mejora. Una de ellas es la refactorización del archivo `main.py`, que actualmente concentra demasiada lógica, dificultando la adaptabilidad; se propone modularizar aún más su estructura para facilitar la escalabilidad y el mantenimiento.

Asimismo, se plantea incorporar funciones físicas no clonables (PUF) para generar llaves criptográficas únicas sin almacenar datos sensibles en memoria, mitigando riesgos de clonación y robo de identidad. También se propone integrar esta solución en esquemas de actualización segura OTA, particularmente en escenarios distribuidos como los de la Industria 5.0, gracias a la portabilidad de su diseño modular y orientado a objetos.

Finalmente, se propone evaluar la eficiencia del sistema bajo distintas condiciones operativas, como variaciones en el tamaño del firmware, número de módulos cifrados y tiempos de respuesta en redes con conectividad limitada. Estas pruebas permitirán ajustar parámetros de diseño y validar la escalabilidad de la solución en entornos reales.

VIII. Agradecimientos

Agradecemos a SECIHTI por el apoyo financiero brindado mediante becas de posgrado, al INAOE por proporcionar un entorno académico de excelencia, y al cuerpo docente del programa de Maestría en Ciencias y Tecnologías de Seguridad por su valiosa formación. Reconocemos especialmente al profesor de la asignatura de *Internet de las Cosas* por su orientación, así como a los autores y desarrolladores de recursos técnicos utilizados en este proyecto.

Referencias

[1] Feng, Yunlong, Linhai Liu, Dong Liang, Qian Chen, Yongchuan Zhou, Zhihao Wang y Linhai Wu: *IBCEB: A bi-authentication Secure Boot Scheme for IoT device based on IBC*. En *2023 3rd International Conference on Electronic Information Engineering and Computer Science (EIECS)*, páginas 1314–1318, 2023.

[2] Kumar, Sudeendra K., Sauvagya Sahoo, Krishna Kiran, Ayas Kanta Swain y K.K. Mahapatra: *A Novel Holistic Security Framework for In-Field Firmware Updates*. En *2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, páginas 261–264, 2018.

[3] Molcut, Alex Ionut, Septimiu Lica y Ioan Lie: *Cybersecurity for Embedded Systems: A review*. 2022 International Symposium on Electronics and Telecommunications (ISETC), 2022.

[4] Wang, Rui y Yonghang Yan: *A Survey of Secure Boot Schemes for Embedded Devices*. En *2022 24th International Conference on Advanced Communication Technology (ICACT)*, páginas 224–227, 2022.

[5] Younis, Mohamed, Mohammad Ebrahimabadi, Suhee Sanjana Mehjabin, Emily Pozniak, Tamim Sookoor y Naghmeh Karimi: *LiSB: Lightweight Secure Boot and Attestation Scheme for IoT and Edge Devices*. IEEE Transactions on Information Forensics and Security, páginas 1–1, 2025.

Diseño e implementación de un sistema colaborativo de verificación de datos segmentados en la nube

C.Angulo-Domínguez
CINVESTAV Unidad Guadalajara
Zapopan 45019, México
cecilia.angulo@cinvestav.mx

A.Díaz-Pérez
CINVESTAV Unidad Guadalajara
Zapopan 45019, México
adiaz@cinvestav.mx

J.L.González-Compeán
CINVESTAV Unidad Tamaulipas
Ciudad Victoria 87130, México
joseluis.gonzalez@cinvestav.mx

***Resumen*—En la actualidad, el almacenamiento de información en la nube se ha convertido en una práctica común tanto en entornos personales como empresariales. No obstante, esta modalidad implica una delegación de la custodia de los datos a terceros, lo que plantea desafíos importantes en términos de confianza, integridad y seguridad. Una de las principales preocupaciones es la posibilidad de que los datos almacenados puedan ser alterados o eliminados sin conocimiento del propietario, lo cual compromete no solo la disponibilidad, sino también la veracidad de la información. Para enfrentar esta problemática, se ha desarrollado el esquema Provable Data Possession (PDP), el cual permite verificar que los datos almacenados en un servidor remoto se mantienen íntegros sin necesidad de descargarlos en su totalidad. En este artículo se propone la implementación de una arquitectura escalable del esquema PDP dentro de una infraestructura basada en servicios de Amazon Web Services (AWS). La propuesta incluye tres componentes principales: el cliente, el servidor y el verificador. Este trabajo se encuentra en una etapa preliminar de consolidación, por lo tanto, los resultados presentados corresponden principalmente a la validación funcional del sistema y en las proyecciones de rendimiento que se esperan alcanzar en pruebas experimentales en etapas posteriores.**

***Palabras clave*—provable data possession, integridad de datos, seguridad en la nube, arquitectura escalable, computación en la nube, verificación remota**

I. Introducción

Con el auge de los servicios de computación en la nube, organizaciones y usuarios individuales han adoptado soluciones de almacenamiento remoto como una alternativa flexible, escalable y económica frente a los métodos tradicionales. Sin embargo, esta delegación del almacenamiento a servicios externos conlleva riesgos inherentes relacionados con la pérdida de control sobre los datos, entre ellos, la manipulación no autorizada, la eliminación maliciosa o accidental, y la falta de garantías sobre la persistencia e integridad de la información [1].

En respuesta a estas inquietudes, se han propuesto diversos mecanismos criptográficos que permiten garantizar la integridad de los datos almacenados [2]–[4]. Uno de los más representativos es el esquema de Provable Data Possession (PDP), que permite al propietario verificar, de manera eficiente, que sus datos permanecen íntegros en un servidor remoto, sin requerir la descarga completa de los archivos. Este esquema se basa en la generación de pruebas criptográficas que el servidor debe responder correctamente, demostrando así que aún conserva los datos originales [5].

La arquitectura planteada busca mostrar la viabilidad de incorporar mecanismos de verificación de integridad de datos en entornos de nube de manera eficiente y escalable. Se enfatiza la importancia de la distribución de funciones en distintos nodos y del aprovechamiento de servicios administrados, lo que simplifica tanto el despliegue como la operación del sistema. La propuesta no solo se presenta como una alternativa a los modelos tradicionales de almacenamiento, sino también como una base para el desarrollo de esquemas auditables y seguros orientados a entornos distribuidos.

Además, la propuesta tiene aplicaciones en escenarios industriales donde la confiabilidad de los datos es crítica. Ejemplos incluyen la trazabilidad en cadenas de suministro, la protección de gemelos digitales en manufactura avanzada, la gestión segura de historiales clínicos en entornos de salud, la verificación de registros en sistemas financieros distribuidos y el resguardo de bitácoras en infraestructuras críticas como energía y transporte. Estos casos de uso evidencian el potencial de la solución para fortalecer la confianza digital en el marco de la Industria.

II. Computación en la nube y provable data possession: Preliminares

En esta sección se presentan los fundamentos conceptuales que sustentan la solución propuesta, abarcando el esquema de Provable Data Possession (PDP), las primitivas criptográficas que lo respaldan y los principios de computación en la nube donde se despliega la arquitectura.

A. Provable Data Possession

El esquema de Provable Data Possession fue introducido formalmente por Ateniese et al. [5] como un mecanismo que permite a un cliente verificar que sus datos se encuentran íntegros en un servidor de almacenamiento remoto, sin necesidad de descargar el archivo completo.

En términos generales, un esquema PDP se basa en un protocolo de desafío y respuesta entre el cliente y el servidor. El cliente divide los archivos en bloques, genera metadatos para cada bloque y los almacena junto con los datos en el servidor. Posteriormente, puede emitir desafíos aleatorios sobre ciertos bloques, y el servidor debe responder con pruebas

válidas de posesión de esos segmentos. Si las respuestas son correctas, se asume con alta probabilidad que los datos permanecen sin alteraciones.

Este enfoque ha sido adaptado y optimizado por diversos trabajos [6]–[8]. Reyes et al. [9] proponen una arquitectura modular basada en building blocks y patrones Manager/Worker, que mejora la eficiencia del PDP mediante contenedores virtuales desplegados en paralelo, permitiendo realizar verificaciones concurrentes en escenarios reales, como el almacenamiento de imágenes satelitales.

B. Funciones Hash Criptográficas

Una función hash criptográfica es un algoritmo que transforma una entrada arbitraria en una salida de longitud fija. Las funciones hash poseen propiedades fundamentales como [10]:

- Resistencia a colisiones: es extremadamente difícil hallar dos entradas diferentes que generen el mismo valor hash. Estas propiedades garantizan la integridad y autenticidad de los datos almacenados.
- Resistencia a preimagen: resulta prácticamente imposible encontrar una entrada que produzca un valor hash dado.
- Resistencia a segunda preimagen: dadas dos entradas distintas, es inviable encontrar otra entrada que genere el mismo hash que una entrada específica.
- Determinismo y eficiencia computacional.

C. Firmas Digitales RSA

Las firmas digitales garantizan la autenticidad y no repudio de los datos transmitidos. En el contexto de PDP, su funcionamiento se puede resumir en:

- Generación de un par de claves: privada (sk) y Pública (pk)
- Firma de los datos (o sus hashes) con la clave privada.
- Verificación de la firma mediante la clave pública.

Cada segmento del archivo es firmado digitalmente antes de ser enviado al servidor. Posteriormente, durante una verificación, el servidor debe proporcionar el segmento firmado, y el verificador podrá comprobar su autenticidad utilizando la clave pública correspondiente. Esta técnica asegura que los datos no han sido modificados y que provienen de una fuente confiable.

Esta técnica también ha sido utilizada en trabajos como el de Li et al. [11], quienes proponen IntegrityChain, una extensión de PDP para entornos descentralizados con blockchain, donde las firmas se combinan con mecanismos de consenso para validar la integridad sin necesidad de confianza en una única entidad.

D. Modelo de Computación en la Nube

El paradigma de la computación en la nube ha transformado significativamente la manera en que las organizaciones acceden, gestionan y escalan sus recursos computacionales. Este modelo permite el acceso remoto, bajo demanda y altamente escalable a servicios de infraestructura, almacenamiento y procesamiento, lo cual reduce costos operativos y mejora la eficiencia del despliegue de sistemas distribuidos. En este contexto, plataformas de nube como Amazon Web Services (AWS) ofrecen un conjunto robusto de servicios que permiten la construcción flexible de arquitecturas orientadas a la seguridad, alta disponibilidad y rendimiento [12]. Entre los servicios para entornos de procesamiento distribuido se destacan:

- EC2: proporciona capacidad de cómputo escalable en la nube, permitiendo el despliegue de máquinas virtuales configurables.
- EFS y S3: permiten el almacenamiento de objetos y archivos de manera persistente y compartida. Mientras que S3 es ideal para almacenar grandes volúmenes de datos no estructurados con alta durabilidad, EFS permite el acceso simultáneo a archivos por múltiples instancias EC2, facilitando la cooperación entre componentes distribuidos.
- IAM: proporciona un control granular sobre el acceso a los recursos de AWS, mediante la definición de políticas y roles. Esto garantiza que solo los agentes autorizados puedan interactuar con los servicios, fortaleciendo así la seguridad del sistema.

El uso coordinado de estos servicios facilita la implementación de sistemas seguros y escalables en entornos cloud, especialmente en arquitecturas donde intervienen múltiples actores y procesos sensibles a la integridad y confidencialidad de la información.

III. Diseño e implementación del sistema PDP

Esta sección describe el diseño e implementación de los componentes que conforman el sistema basado en el esquema Provable Data Possession, desplegado sobre infraestructura en la nube. El sistema está dividido en tres entidades principales: cliente, servidor y verificador, cada una con funciones específicas que en conjunto permiten asegurar la integridad de los datos almacenados.

A. Cliente

El Cliente se desempeña como la entidad propietaria de los datos y es responsable de preparar la información antes de su almacenamiento remoto. Para ello, ejecuta localmente una serie de operaciones previas al envío, tales como la fragmentación de archivos, la generación de metadatos y la aplicación de funciones criptográficas. Estas tareas aseguran que los datos estén correctamente estructurados y protegidos antes de ser transferidos al Servidor. Entre sus funciones principales se encuentran:

- Generar un par de claves criptográficas RSA (clave pública y privada).
- Dividir los archivos en segmentos.
- Calcular el hash SHA-256 de cada segmento.
- Firmar digitalmente cada segmento utilizando RSA.
- Enviar al servidor los segmentos junto con su firma y metadatos asociados.
- Almacenar localmente los metadatos
- Solicitar la verificación de integridad de los segmentos.

En la Figura 1 se representa gráficamente el flujo de operaciones ejecutadas por el cliente. Se puede observar cómo,

tras la segmentación del archivo, cada fragmento se somete a un proceso de hash y firma.

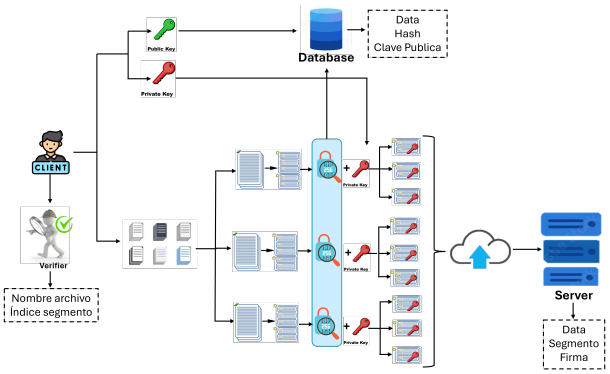


Fig. 1. Representación del componente cliente.

El algoritmo 1 detalla el procedimiento que asegura que cada fragmento enviado al servidor puede ser autenticado posteriormente mediante su firma y verificado con su hash.

Algorithm 1 Proceso del Cliente

```
cPrivada, cPublica ← GeneraClavesRSA()
BD ← ConectarSQLite()
for cada archivo en la lista do
    segmentos ← DividirArchivo(archivo)
    for cada segmento en segmentos do
        hash ← SHA256(segmento)
        firma ← Firmar(cPrivada, segmento)
        AlmacenaMetadato(data, hash, cPublica)
        EnviarAlServidor(data, segmento, firma)
    end for
end for
SolicitarVerificacion(data)
```

B. Servidor

El servidor es responsable de recibir los datos del cliente y almacenarlos de manera segura. Para cada segmento recibido, conserva tanto el contenido como sus metadatos, permitiendo posteriormente su recuperación para procesos de verificación. El servidor también actúa como intermediario entre el cliente y el verificador, respondiendo a solicitudes de verificación mediante la entrega de los segmentos requeridos y sus respectivos metadatos. Sus funciones principales incluyen:

- Recibir los segmentos firmados desde el cliente.
- Asociar cada segmento con su metadato.
- Almacenar los datos de manera organizada.
- Atender solicitudes del verificador enviando los segmentos requeridos junto con sus metadatos.

En la Figura 2 se ilustra el comportamiento del servidor. Se muestra cómo recibe de forma continua los paquetes enviados por el cliente y los almacena, manteniéndolos disponibles para responder las peticiones del verificador.

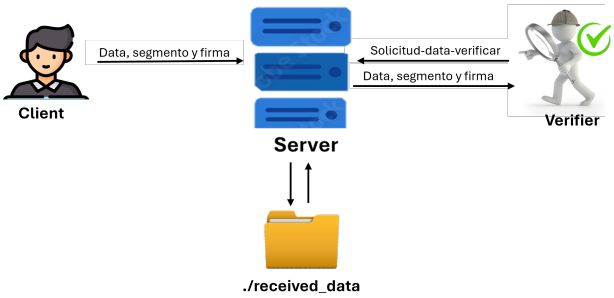


Fig. 2. Representación del componente servidor.

En el Algoritmo 2, el servidor desempeña el rol de un repositorio pasivo de datos, encargado únicamente de almacenar los segmentos recibidos, carece de la capacidad para alterar la información sin ser detectado, ya que cualquier modificación comprometería la verificación de integridad realizada posteriormente por el verificador.

Algorithm 2 Proceso del Servidor

```
while Servidor activo do
    (origen, peticion) ← EsperarPeticion()
    if origen = Cliente then
        (data, segmento, firma) ← peticionPost
        Almacenar(data, segmento, firma)
    else if origen = Verificador then
        (data, segmento, firma) ← peticionGet
        EnviarVerificador(data, segmento, firma)
    end if
end while
```

C. Verificador

El verificador es una entidad encargada de comprobar la autenticidad e integridad de los segmentos almacenados en el servidor. Este proceso se inicia a petición del cliente, quien determina qué bloques deben ser verificados. El verificador solicita al servidor los segmentos correspondientes y, una vez recibidos, valida sus firmas digitales utilizando la clave pública del cliente. Posteriormente, calcula los valores hash de los segmentos y los compara con los valores de referencia previamente almacenados, asegurando que los datos no hayan sido alterados. Las funciones principales del verificador son:

- Solicitar los segmentos designados por el cliente.
- Verificar las firmas digitales empleando la clave pública del cliente.
- Calcular y comparar los hashes para validar la integridad de los datos.

La Figura 3 representa gráficamente el ciclo de verificación, desde la generación del desafío hasta la validación de la integridad de los datos. La figura 3 evidencia cómo el verificador accede a un subconjunto de segmentos sin necesidad de recuperar todo el archivo, reforzando así la eficiencia del esquema PDP.

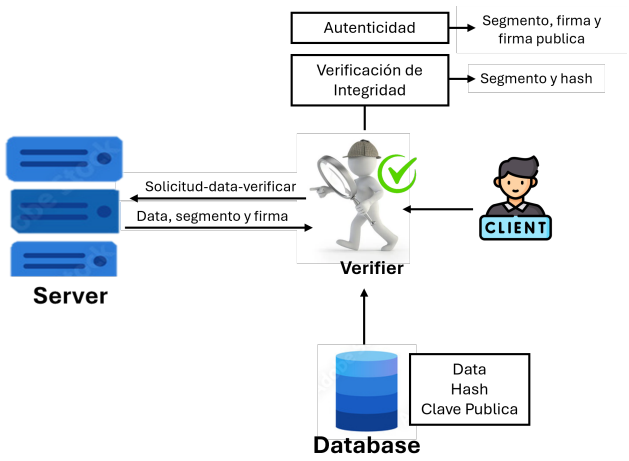


Fig. 3. Representación del componente verificador.

El Algoritmo 3 valida tanto la autenticidad como la integridad de los datos recibidos. Mediante el uso de firma digital y hash garantiza que incluso una mínima alteración en los segmentos será detectada con alta probabilidad.

Algorithm 3 Proceso del Verificador

```
segmentos ← PETICIONCLIENTE()
segmentosRecib ← SOLICITARSEGMENTOSSERVIDOR()
for cada segmento recibido do
    autenticidad ← VERIFICAFIRMA(segmento, firma)
    integridad ← VERIFICAHASH(segmento, hash)
end for
```

IV. Arquitectura del sistema en la nube

La arquitectura propuesta para la implementación del esquema Provable Data Possession se despliega sobre una infraestructura en la nube utilizando servicios de Amazon Web Services. Esta arquitectura se diseñó con base en principios de modularidad y escalabilidad, permitiendo separar los componentes funcionales del sistema (cliente, servidor y verificador) en diferentes instancias o servicios, y facilitar así su despliegue, mantenimiento y evolución.

A. Descripción general de la arquitectura

La arquitectura se compone de tres entidades lógicas principales:

- Cliente: desplegado en una instancia EC2, se encarga de realizar la segmentación de los datos, generar las firmas digitales correspondientes y transferir los segmentos al servidor.
- Servidor: implementado sobre una instancia EC2, actúa como repositorio de los datos segmentados. Almacena los bloques firmados y responde a las solicitudes del verificador sin capacidad de modificar los datos sin ser detectado.
- Verificador: ejecutado en una instancia EC2, recibe del cliente la instrucción de verificar determinados segmentos. Solicita dichos segmentos al servidor y valida su autenticidad mediante la verificación de firmas y el cálculo de hashes.

B. Diagrama de la arquitectura en la nube

El sistema se estructura en tres niveles funcionales (ver Figura 4), cada uno correspondiente a uno de los roles del esquema Provable Data Possession (PDP). La infraestructura ha sido desplegada sobre servicios de Amazon Web Services (AWS), utilizando Terraform como herramienta de aprovisionamiento y automatización. Esta elección permite gestionar la infraestructura como código, garantizando la reproducibilidad, escalabilidad y trazabilidad de las configuraciones, además de facilitar el control de versiones y la gestión de cambios en el entorno. El flujo de operación se describen a continuación:

- Inicialización:
  - El cliente genera su propio par de claves RSA para firmar sus datos.
  - El servidor permanece disponible y expone un endpoint para recibir segmentos y metadatos.
- Carga de datos:
  - El cliente divide el archivo en segmentos, calcula sus hashes, firma cada uno y envía los segmentos y algunos metadatos firmados al servidor, mientras que los hashes y metadatos se guardan en un sistema de almacenamiento compartido (EFS) accesible por clientes y verificadores.
  - El servidor almacena los segmentos de datos y metadatos recibidos por el cliente localmente.
- Verificación:
  - A solicitud del cliente, el verificador inicia un proceso de validación sobre un subconjunto aleatorio de segmentos.
  - El verificador solicita dichos segmentos al servidor, quien los recupera desde su almacenamiento local.
  - Utilizando las firmas y hashes almacenados en EFS, el verificador comprueba la integridad y autenticidad de los datos.
  - Los resultados de la verificación se registran, y en caso de detectar inconsistencias, se generan alertas automáticas.
- Escalabilidad y replicación:
  - Varios clientes y verificadores pueden operar simultáneamente sobre la infraestructura compartida, permitiendo múltiples procesos de carga, consulta y verificación en paralelo sin interferencias entre sí.
  - Las instancias EC2 del servidor pueden escalar horizontalmente según la demanda.
  - La replicación de datos en S3 y EFS garantiza la disponibilidad ante fallos de zona o región, asegurando la continuidad del servicio y la integridad de la información almacenada.

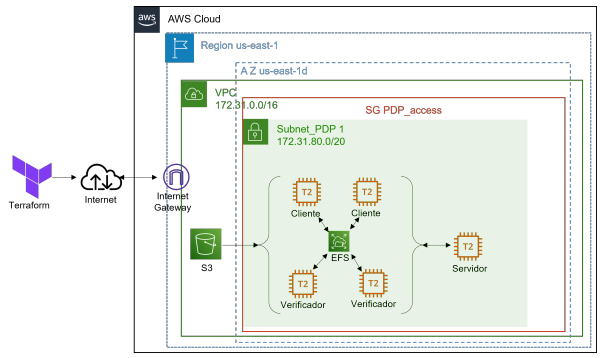


Fig. 4. Arquitectura del esquema PDP en la nube.

V. Resultados esperados y pruebas de concepto

La presente sección tiene como objetivo exponer los resultados esperados derivados de la implementación del esquema Provable Data Possession en un entorno de computación en la nube, así como describir las pruebas de concepto iniciales realizadas para validar su funcionamiento a nivel lógico, criptográfico y operativo. Dado que este trabajo se encuentra en fase de consolidación arquitectónica, los resultados aquí descritos se enfocan en la validación funcional del sistema y en las proyecciones de rendimiento que se esperan alcanzar en pruebas experimentales futuras.

A. Resultados esperados

El sistema propuesto fue diseñado bajo principios de seguridad y modularidad. En este sentido, los resultados esperados se agrupan en las siguientes categorías:

- Seguridad de los datos.
  - Integridad garantizada: gracias al uso de funciones hash SHA-256 y firmas digitales RSA, se espera que cualquier alteración en los datos pueda ser detectada con alta certeza, incluso al verificar solo un subconjunto aleatorio de segmentos.
  - Autenticidad de origen: la verificación mediante la clave pública permite confirmar que los datos provienen efectivamente del cliente autorizado.
  - Sin necesidad de copia local: el cliente puede eliminar los archivos originales una vez almacenados, manteniendo solo los metadatos.
- Eficiencia y escalabilidad
  - Bajo consumo de ancho de banda: el sistema permite verificar archivos completos mediante el análisis de unos pocos bloques, evitando transferencias masivas de datos.
  - Escalabilidad horizontal: el sistema está diseñado para permitir múltiples clientes y verificadores operando en paralelo sin interferencias.
  - Procesamiento distribuido: al separar los roles funcionales en instancias EC2 distintas, se espera un aprovechamiento más equilibrado de los recursos computacionales.

B. Pruebas de concepto realizadas

Se llevaron a cabo pruebas de concepto sobre la arquitectura implementada en AWS con el fin de verificar la coherencia entre los módulos y la viabilidad del flujo operativo. Estas pruebas no se enfocaron en rendimiento cuantitativo, sino en la validación funcional.

Prueba 1: Flujo completo de un archivo simple.
Objetivo: verificar que el ciclo completo(cliente, servidor, verificador) se realiza correctamente con un archivo de prueba.

- Acciones realizadas:
  - Segmentación y firma de un archivo de 5 MB en el cliente.
  - Almacenamiento del archivo segmentado en el servidor.
  - Solicitud del cliente al verificador la comprobación de 3 bloques.
  - Solicitud de obtención de 3 bloques del verificador al servidor para su validación.
  - Verificación exitosa de firmas y hashes.

Resultado: flujo completado sin errores y con tiempos de respuesta aceptables ($< 2\,$s por segmento).

Prueba 2: Detección de corrupción intencionada.
Objetivo: comprobar la eficacia del sistema ante la modificación maliciosa de un bloque.

- Acciones realizadas:
  - Se alteró manualmente uno de los segmentos en el almacenamiento del servidor.
  - El verificador solicitó el bloque modificado.
  - La verificación de firma y hash falló.

Resultado: el sistema detectó correctamente la modificación, confirmando la funcionalidad de las garantías de integridad.

Prueba 3: Operación concurrente básica.
Objetivo: simular la operación de múltiples clientes/verificadores en paralelo.

- Acciones realizadas:
  - Dos clientes ejecutaron simultáneamente envíos de archivos distintos.
  - Un verificador consultó bloques de ambos conjuntos en paralelo.
  - Las respuestas fueron gestionadas correctamente por el servidor.

Resultado: la arquitectura soportó la concurrencia básica sin colisiones ni errores, validando su diseño modular.

Aunque las pruebas de concepto validaron la funcionalidad del sistema, su alcance sigue siendo limitado. Será necesario en futuras fases incorporar métricas de desempeño más detalladas, tiempos de verificación, latencia, uso de recursos, ancho de banda y costos en AWS, así como ensayos de escalabilidad con múltiples clientes, archivos de gran tamaño y condiciones de red diversas.

VI. Conclusiones y trabajos futuros

En esta sección se presentan las conclusiones derivadas del desarrollo e implementación del sistema propuesto, así como una reflexión sobre sus aportaciones técnicas.

En este trabajo se presentó el diseño e implementación de un sistema basado en el esquema Provable Data Possession, desplegado sobre una arquitectura modular en la nube utilizando servicios de AWS. El objetivo principal del sistema es permitir la verificación eficiente de la integridad de los datos almacenados en servidores remotos, sin requerir su descarga completa, utilizando mecanismos criptográficos como funciones hash y firmas digitales.

La propuesta se estructuró en tres componentes principales: cliente, servidor y verificador, los cuales interactúan de forma autónoma y distribuida para garantizar la seguridad y disponibilidad de los datos. Las pruebas de concepto realizadas permitieron validar el flujo operativo, la detección de modificaciones en los datos y la correcta separación de responsabilidades entre componentes.

Entre los beneficios más relevantes de la solución se encuentran su escalabilidad horizontal, la eficiencia en el uso de recursos, y la posibilidad de adaptarse a distintos entornos y políticas de seguridad. Además, la arquitectura propuesta proporciona una base sólida para extender el sistema hacia entornos más complejos o regulados.

Un aspecto crítico identificado en esta fase es la gestión de claves criptográficas. Actualmente, la generación y resguardo de las claves RSA recae en el cliente, lo cual implica riesgos en caso de pérdida, compromiso o mal manejo. La ausencia de mecanismos de rotación de claves, recuperación segura y almacenamiento en módulos de seguridad hardware (HSM) representa una vulnerabilidad que deberá ser atendida en el futuro para asegurar la solidez del sistema.

La propuesta contribuye directamente a los pilares de la Industria 5.0 al fortalecer la confianza digital en servicios de almacenamiento remoto, incrementar la resiliencia frente a ataques mediante verificaciones criptográficas distribuidas, y favorecer la soberanía tecnológica al garantizar que los datos permanezcan bajo control verificable del cliente, incluso cuando son gestionados por proveedores externos.

Trabajos futuros

Como parte del trabajo a futuro, se contemplan diversas líneas de mejora e investigación:

- Evaluación experimental a gran escala: se proyecta realizar pruebas sistemáticas en escenarios con múltiples clientes y verificadores concurrentes, manejo de archivos de gran tamaño (del orden de gigabytes o terabytes), y despliegues distribuidos en nodos heterogéneos dentro de Kubernetes y diferentes regiones de nube.
- Mejora de la gestión de claves: se planea incorporar mecanismos de rotación de claves, recuperación segura y uso de módulos de seguridad hardware (HSM), con el fin de mitigar riesgos derivados del mal manejo o compromiso de las claves criptográficas.
- Se explorará la aplicación de Zero-Knowledge Proofs(ZKP) para mejorar la privacidad en la validación de datos.
- Incorporación de algoritmos alternativos: se analizará el uso de esquemas criptográficos más ligeros, como ECC y BLS, que podrían reducir la sobrecarga computacional frente al uso de RSA y SHA-256 en escenarios de alta concurrencia.
- Evaluación en entornos regulados: se plantea validar el sistema bajo normativas internacionales de ciberseguridad en la nube, tales como ISO/IEC 27017, ISO/IEC 27018, la directiva NIS2 y las guías de seguridad del NIST.

Referencias

[1] K. Zhu, Y. Ren, and Q. Zhu, "A provable data possession protocol in cloud storage systems with fault tolerance," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, 2021, pp. 1–6.

[2] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1788–1797, 2020.

[3] N. Kaaniche, M. Laurent, and S. Canard, "Cooperative set homomorphic proofs for data possession checking in clouds," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 102–117, 2021.

[4] M. Tian, Z. Xie, H. Zhong, and Z. Chen, "Lightweight proofs of storage with public verifiability from lattices," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2020, pp. 551–558.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 598–609. [Online]. Available: https://doi.org/10.1145/1315245.1315318

[6] S. OGISO, M. MOHRI, and Y. SHIRAISHI, "Transparent provable data possession scheme for cloud storage," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–5.

[7] S. Dong, J. Jiao, and S. Li, "A multiple-replica provable data possession algorithm based on branch authentication tree," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020, pp. 400–404.

[8] X. A. Wang and R. Li, "Improved secure provable data possession scheme for cloud storage," in *2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, 2022, pp. 456–460.

[9] H. G. Reyes-Anastacio, J. Gonzalez-Compean, M. Morales-Sandoval, and J. Carretero, "A data integrity verification service for cloud storage based on building blocks," in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, 2018, pp. 201–206.

[10] V. Mulder, A. Mermoud, V. Lenders, and B. Tellenbach, *Hash Functions*. Springer Nature, 2023.

[11] Y. Li, Y. Yu, R. Chen, X. Du, and M. Guizani, "Integritychain: Provable data possession for decentralized storage," *IEEE Journal on Selected Areas in Communications*, vol. PP, pp. 1–1, 04 2020.

[12] S. Shrivastava, N. Srivastav, A. Artasanchez, I. Sayed, and S. D, *AWS for Solutions Architects: The definitive guide to AWS Solutions Architecture for migrating to, building, scaling, and succeeding in the cloud*. Packt Publishing, 2023. [Online]. Available: https://books.google.com.mx/books?id=x6W7EAAAQBAJ

Manejo de políticas para la evaluación eficiente de solicitudes en un modelo de control de acceso basado en atributos

R.A.Ibarra-García
CINVESTAV Unidad Guadalajara
Zapopan 45019, México
ricardo.ibarra@cinvestav.mx

A.Díaz-Pérez
CINVESTAV Unidad Guadalajara
Zapopan 45019, México
adiaz@cinvestav.mx

J.L.González-Compeán
CINVESTAV Unidad Tamaulipas
Ciudad Victoria 87130, México
joseluis.gonzalez@cinvestav.mx

***Resumen*—El control de acceso en sistemas distribuidos, especialmente en el contexto de la Industria 5.0, requiere mecanismos eficientes y adaptables para garantizar que solo los usuarios autorizados puedan interactuar con recursos críticos bajo condiciones específicas. Sin embargo, los enfoques tradicionales como el control de acceso basado en roles o las listas de control de acceso tienden a ser limitados en expresividad y poco eficientes al evaluar grandes volúmenes de políticas contextuales, dificultando su escalabilidad y capacidad de adaptación. En este artículo se presenta un modelo dinámico de control de acceso basado en eventos, donde cada uno se define como una conjunción de atributos: sujeto, recurso, espacio, tiempo y acción. Se implementó un prototipo funcional del sistema y se validó la propuesta mediante un estudio de caso simulado, utilizando políticas construidas a partir del dataset *KDD Cup 1999*. Las pruebas preliminares mostraron un comportamiento consistente ante variaciones en los atributos de las solicitudes, demostrando así la aplicabilidad del modelo.**

***Palabras clave*—control de acceso, industria 5.0, sistemas distribuidos, políticas contextuales**

I. Introducción

En el contexto de la Industria 5.0 [1] y los sistemas distribuidos, el control de acceso cumple una función central en la gestión de recursos digitales sensibles frente a accesos no autorizados [2]. La complejidad de estos entornos, impulsada por el Internet de las Cosas (*IoT*), la inteligencia artificial y la interacción colaborativa entre humanos y máquinas, demanda mecanismos de control de acceso dinámicos, granulares y adaptables a condiciones contextuales cambiantes [3].

Los enfoques tradicionales como las Listas de Control de Acceso (*ACL's*) y el Control de Acceso Basado en Roles (*RBAC*) (ver Figura 1) presentan limitaciones para responder a escenarios dinámicos [4], [5].

Aunque han sido ampliamente utilizados por su facilidad administrativa, no permiten incorporar atributos cambiantes ni considerar múltiples dimensiones contextuales [6]. Esto resulta problemático cuando las decisiones de acceso deben tomarse en función de condiciones

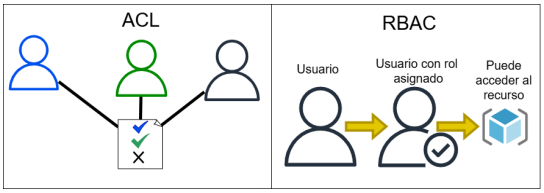


Figura 1. Modelos ACL y RBAC.

variables como la ubicación, el horario o el recurso [7]. Por ejemplo, un esquema RBAC tradicional puede otorgar acceso a un ingeniero de red para administrar servidores, pero no distingue si el intento ocurre fuera del horario laboral o desde una ubicación no autorizada.

En respuesta a estas limitaciones el Control de Acceso Basado en Atributos (*ABAC*) ofrece una alternativa flexible al considerar atributos del sujeto, recurso, acción y contexto espacio-temporal [8]. Se ha mostrado eficaz en entornos heterogéneos y dinámicos como infraestructuras críticas, sistemas de salud, ciudades inteligentes e industrias conectadas [9]–[11].

Sin embargo, el uso de *ABAC* no está libre de desafíos, especialmente en cuanto a la eficiencia de evaluación de políticas cuando el volumen de solicitudes y la cantidad de atributos involucrados crece.

En este artículo se presentan las siguientes contribuciones:

- Un modelo dinámico de control de acceso basado en eventos, en el cual cada solicitud se representa como una conjunción estructurada de atributos clave: sujeto, recurso, espacio, tiempo y acción.
- Diseño de una arquitectura modular para la evaluación eficiente de políticas de acceso y adaptabilidad del sistema frente a contextos variables.

Para validar esta propuesta, se desarrolló un prototipo funcional y se evaluó su comportamiento utilizando un estudio de caso basado en el conocido dataset de

intrusiones de red *KDD Cup 1999* [12], un benchmark clásico utilizado en la detección de intrusiones y la investigación en ciberseguridad.

El resto del artículo se desarrolla de la siguiente manera: En la Sección II se presentan los trabajos relacionados que abordan nuestro problema, en la Sección III se describe el modelo de control de acceso basado en atributos orientado a eventos, en la Sección IV se muestran los resultados preeliminares que se obtuvieron a partir del mapeo del dataset *KDD Cup*. En la Sección V presentamos las conclusiones obtenidas y finalmente en la Sección VI se presenta el trabajo futuro de la investigación.

II. Trabajo relacionado

En esta sección se presenta una revisión de trabajos representativos que abordan directamente el problema de la evaluación eficiente y el ordenamiento lógico o estructural de políticas ABAC. Estos criterios permiten contrastar nuestra propuesta con modelos estructurales, optimizaciones algorítmicas y enfoques descentralizados.

Yang *et al.* [13] introduce el modelo G_{ABAC}, una representación de políticas ABAC mediante grafos dirigidos que permiten evaluar accesos como flujos entre nodos (usuarios y recursos). Implementado en la base de datos *Neo4j*, este enfoque posibilita el uso de consultas tipo Cypher para analizar y aplicar políticas. Los autores reportan que, para conjuntos de hasta 10,000 nodos y 50,000 relaciones, la latencia promedio en la evaluación de políticas se mantuvo por debajo de los 20 ms, incluso en escenarios con múltiples condiciones contextuales. Además, se observó que el tiempo de análisis de políticas complejas se redujo en un 65 % respecto a sistemas tradicionales basados en árboles.

Liu *et al.* [14] proponen una técnica de recuperación de políticas que mejora la eficiencia en ambientes con miles de reglas y múltiples atributos. El método utiliza identificadores binarios para filtrar por atributo y un índice de profundidad para ordenar por valores. Las pruebas se realizaron con hasta 10,000 políticas y mostraron que el enfoque propuesto fue entre 3 y 5 veces más rápido que los métodos tradicionales de evaluación secuencial. Además, se mantuvo una precisión del 100 % en la recuperación de políticas válidas, demostrando que la optimización no comprometía la exactitud.

Además, Paul *et al.* [15] exploran la utilización de estructuras de datos de alta dimensión, en particular los árboles C-ND, para indexar eficientemente atributos continuos y discretos en políticas ABAC. La propuesta permite ejecutar búsquedas por similitud (útil en accesos de emergencia) de forma eficiente. En sus experimentos, realizados sobre datasets con más de 50 atributos por política, lograron reducir los tiempos de búsqueda en un 73 % comparado con enfoques sin indexación. Además, la tasa de decisiones correctas ante solicitudes ambiguas alcanzó el 96.8 %, mencionando que valida su uso en contextos de tolerancia a incertidumbre.

Así mismo, Hu *et al.* [16] proponen una arquitectura descentralizada para puntos de decisión de políticas ABAC utilizando blockchain, mediante un mecanismo optimizado llamado EBEC (Echo-Based Execution Conclude). Su enfoque reduce la redundancia en la evaluación y mejora la eficiencia mediante técnicas como Echo Compacting y balanceo de carga entre nodos. Las pruebas en infraestructura AWS demostraron una mejora del 35.6 % en el rendimiento comparado con otros enfoques blockchain, destacando su aplicabilidad en entornos distribuidos.

La Tabla I muestra una comparación cualitativa entre los trabajos seleccionados y la propuesta aquí desarrollada, utilizando cinco criterios para la evaluación eficiente en modelos ABAC. Como puede observarse, todos los enfoques revisados abordan la eficiencia en la evaluación, pero difieren en su estructura interna, capacidad para manejar atributos heterogéneos, soporte a solicitudes ambiguas y escalabilidad. En particular, solo Paul *et al.* considera solicitudes parcialmente coincidentes, mientras que la propuesta de este trabajo incorpora una representación estructurada y un mecanismo de afinidad para ordenar políticas y agilizar la evaluación. A diferencia de otros, el modelo propuesto cumple con los criterios establecidos y resulta eficaz en contextos dinámicos.

III. Modelo ABAC orientado a eventos

Este trabajo propone un modelo de control de acceso dinámico que organiza y evalúa políticas ABAC de forma eficiente mediante un proceso de agrupamiento semántico y evaluación orientada por similitud cuyo proceso se muestra en la Figura 2.

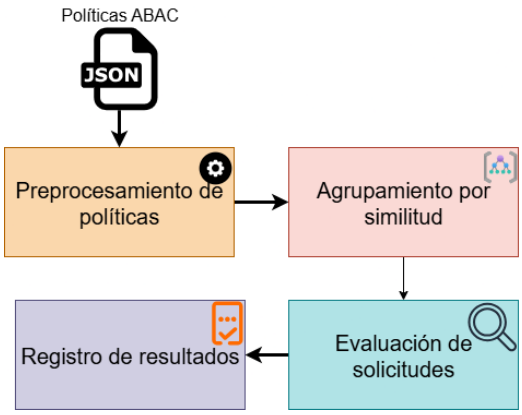


Figura 2. Diagrama conceptual del modelo propuesto.

Cuadro I
MATRIZ DE DOMINANCIA VISUAL: COMPARACIÓN DE TRABAJOS RELACIONADOS.

Leyenda: ✓ Cumple completamente ▲ Parcialmente cumple ✗ No cumple

Artículo	Evaluación eficiente	Modelo estructural	Atributos continuos/discretos	Solicitudes ambiguas	Escalabilidad
Yang *et al.* [13]	✓	✓	▲	✗	✓
Liu *et al.* [14]	✓	▲	✓	✗	✓
Paul *et al.* [15]	✓	▲	✓	✓	✓
Hu *et al.* [16]	✓	✓	✗	✗	✓
Propuesta	✓	✓	✓	✓	✓

La idea central consiste en representar las situaciones de acceso permitidas (eventos) como combinaciones específicas de atributos, y luego aplicar una técnica de agrupación para identificar patrones comunes en dichos eventos.

Un **evento** se define como una tupla de cinco atributos:

$$e = (A_0, A_1, A_2, A_3, A_4)$$

donde cada A_i representa un atributo distintivo del contexto de acceso.

- A_0 corresponde al *sujeto*, por ejemplo la identidad o rol del solicitante
- A_1 es el *recurso* u objeto al que se desea acceder
- A_2 se refiere a la *ubicación* del sujeto o del recurso
- A_3 es el *tiempo* o instante/intervalo temporal del acceso
- A_4 corresponde a la *acción* que el sujeto intenta realizar sobre el recurso.

Cada atributo A_i toma un valor de un dominio predefinido para ese tipo de atributo (por ejemplo, el atributo de sujeto podría tomar valores como `Estudiante`, `Profesor` o `Administrador`).

Una **política de acceso** P se define como un conjunto de eventos, es decir $P = \{e_1, e_2, \ldots, e_n\}$. Cada evento en la política representa una combinación particular de atributos que está permitida por dicha política. De este modo, una política especifica explícitamente los contextos o situaciones (definidos por atributos) en los cuales ciertas acciones están autorizadas.

Cada solicitud de acceso se representa como una tupla de atributos, y se evalúa respecto a un conjunto de políticas, cada una definida por combinaciones particulares de contexto (eventos autorizados). Para evitar evaluaciones exhaustivas sobre todo el conjunto de políticas, el modelo construye un espacio estructurado de eventos donde se identifican regiones de alta similitud contextual. Este agrupamiento se realiza en función de una métrica de similitud ponderada entre eventos, que considera el peso relativo de cada atributo:

$$\text{sim}(e_i, e_j) = \sum_{k=1}^{n} w_k \cdot \delta_k(e_i, e_j) \tag{1}$$

donde w_k representa la importancia relativa del atributo k y δ_k es una función indicadora de coincidencia para dicho atributo. Los pesos se derivan automáticamente del comportamiento estadístico de los datos, asignando mayor peso a atributos más informativos.

Sobre esta base, se genera una estructura de agrupamiento que permite segmentar las políticas en subconjuntos temáticamente coherentes. Al recibir una nueva solicitud, el sistema identifica el subconjunto más afín y evalúa las políticas en un orden descendente de afinidad, evitando la comparación con políticas irrelevantes.

Este mecanismo de organización y evaluación priorizada permite reducir el tiempo de respuesta en entornos dinámicos o distribuidos, donde las políticas pueden ser numerosas y contextuales.

IV. RESULTADOS PREELIMINARES

Para evaluar el comportamiento del modelo propuesto, se realizó un estudio de caso utilizando el dataset *KDD Cup 1999*. Este conjunto de datos, ampliamente utilizado en el campo de la detección de intrusos, contiene descripciones detalladas de conexiones de red simuladas en un entorno controlado.

Dado que los registros de solicitudes de acceso reales suelen estar restringidos por razones de privacidad y confidencialidad, se optó por adaptar el *KDD Cup* como una fuente alternativa. A través de un mapeo contextual, cada conexión registrada fue reinterpretada como un evento compatible con el modelo ABAC, utilizando atributos clave que reflejan el sujeto, el recurso, la ubicación, el tiempo y la acción realizada.

IV-A. Mapeo del dataset KDD Cup a eventos ABAC

El mapeo propuesto transforma los registros del dataset en eventos evaluables bajo el marco ABAC, como se resume en la Tabla II:

Este mapeo preserva los elementos clave de un evento de acceso: *¿Quién?*, *¿Qué?*, *¿Dónde?*, *¿Cuándo?* y

Cuadro II
MAPEO DE ATRIBUTOS ABAC A COLUMNAS DEL DATASET KDD CUP 1999.

Atributo ABAC	Columnas utilizadas	Interpretación
Sujeto (A0)	`logged_in`, `is_guest_login`	Representa al actor de la conexión. Clasificado como *Interno*, *Invitado* o *Externo* según tipo de login.
Recurso (A1)	`service`	Sistema o servicio accedido: ftp, http, smtp, telnet, etc.
Ubicación (A2)	`land`	Determina si el origen es local (land=1) o remoto (land=0).
Tiempo (A3)	`duration`	Duración de la conexión; interpretada como acceso corto, medio o largo.
Acción (A4)	`protocol_type`	Tipo de operación ejecutada, inferida a partir del protocolo (por ejemplo, telnet implica conexión remota).

Acción, derivando sujeto, recurso, acción, ubicación y tiempo de sus atributos originales. Así, el dataset se transforma en una bitácora de solicitudes compatible con políticas ABAC orientadas a eventos, facilitando la evaluación y el diseño de esquemas de control de acceso sobre datos reales o simulados.

IV-B. Evaluación experimental inicial

Se presentan los resultados experimentales iniciales de la exactitud y el rendimiento del sistema de control de acceso bajo diferentes cargas de trabajo. Se utilizó un conjunto fijo de 100 políticas, y se variaron el número de solicitudes evaluadas entre 30 y 300,000 para analizar el comportamiento en términos de eficiencia.

Experimento 1: Exactitud en la evaluación de acceso

En este experimento se evaluó la capacidad del sistema para determinar correctamente las decisiones de acceso (`permit` o `deny`) bajo diferentes condiciones: coincidencias exactas, uso de comodines ($*$) y atributos categóricos como duración. La evaluación se realizó sobre un total de 2039 solicitudes. Las Tablas III y IV muestra los resultados de exactitud obtenidos.

Cuadro III
MÉTRICAS PRINCIPALES OBTENIDAS

Métrica	Valor
Exactitud (Accuracy)	1.0
Precisión (permit)	1.0
Cobertura (Recall, permit)	1.0
F1 Score (permit)	1.0
Precisión (deny)	1.0
Cobertura (Recall, deny)	1.0
F1 Score (deny)	1.0

El sistema mostró un desempeño perfecto en la evaluación de acceso, con decisiones totalmente consistentes respecto a las políticas establecidas. Se alcanzó una exactitud, precisión y cobertura del 100 %, sin falsos positivos ni falsos negativos. Este resultado era esperado, ya que el experimento fue diseñado para validar el mecanismo de evaluación en condiciones controladas: se utilizaron políticas y solicitudes predefinidas, cuidadosamente seleccionadas para representar situaciones bien delimitadas, incluyendo coincidencia exacta, uso de comodines y atributos contextuales. La ausencia de errores refleja la correcta implementación del modelo y su capacidad para operar conforme a las reglas establecidas, más que un comportamiento generalizable a entornos abiertos.

Cuadro IV
MATRIZ DE CONFUSIÓN

	Predicho: permit	Predicho: deny
Esperado: permit	316	0
Esperado: deny	0	1723

Experimento 2: Tiempo de evaluación

La Tabla V muestra los resultados obtenidos (en segundos como unidad de tiempo) para distintos tamaños de lote de solicitudes, indicando el tiempo total de evaluación (TT), el tiempo promedio por solicitud (TP/S) y el *throughput* (solicitudes por segundo).

Cuadro V
RESULTADOS DE RENDIMIENTO DEL SISTEMA DE EVALUACIÓN DE ACCESO

Nº Solicitudes	TT	TP/S	Throughput (sol/s)
30	0.030641	0.000939	984
300	0.247048	0.000735	1214
3,000	0.785019	0.000190	3824
30,000	7.404685	0.000240	4052
300,000	72.694182	0.000235	4128

La Figura 3 presenta el comportamiento del **tiempo promedio de evaluación por solicitud** conforme aumenta el número de solicitudes en el lote, usando una escala logarítmica en el eje X. Se observa una disminución y posterior estabilización del tiempo promedio conforme el volumen de solicitudes crece, alcanzando valores menores a 0.25 ms por solicitud incluso para cargas masivas.

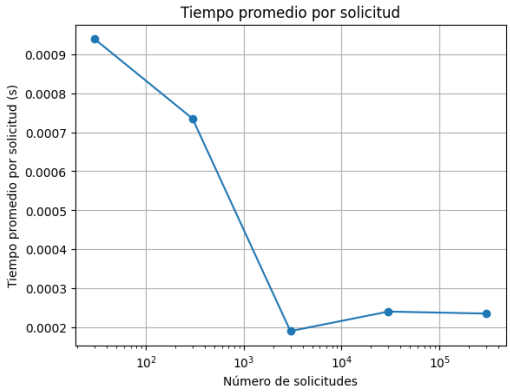


Figura 3. Tiempo promedio de evaluación por solicitud en función del número de solicitudes.

Por su parte, la Figura 4 muestra el **throughput** (solicitudes evaluadas por segundo), que aumenta rápidamente con el tamaño del lote y se estabiliza por encima de 4,000 solicitudes/segundo para cargas mayores a 3,000 solicitudes.

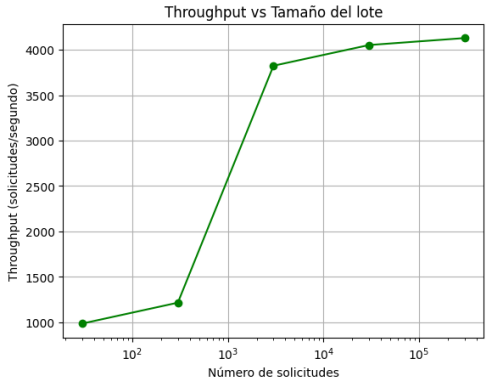


Figura 4. Throughput (solicitudes/seg) respecto al tamaño del lote.

Finalmente, la Figura 5 representa el **tiempo total de evaluación** en función del número de solicitudes, mostrando una relación lineal que evidencia la eficiencia constante por solicitud, incluso bajo cargas masivas.

Los resultados experimentales demuestran que el sistema de evaluación propuesto es eficiente y escalable.

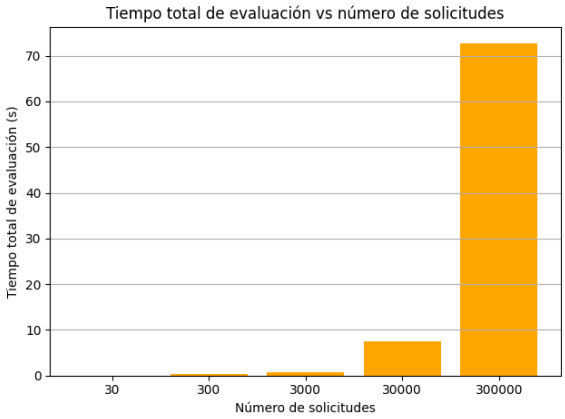


Figura 5. Tiempo total de evaluación en función del número de solicitudes evaluadas.

El tiempo promedio por solicitud se mantiene estable y bajo (menor a 0.25 ms) incluso ante cargas de hasta 300,000 solicitudes. El *throughput* alcanzado supera las 4,000 solicitudes por segundo, validando que el sistema es apto para su uso en escenarios de alta demanda y operación en tiempo real.

Experimento 3: Comparación de modelos

El objetivo de este experimento es demostrar la mejora del modelo propuesto frente al enfoque tradicional de evaluación. Para ello, se diseñó una prueba de escalabilidad basada en la variación del número de políticas (N), utilizando un conjunto fijo de 2000 solicitudes y evaluando cuatro tamaños diferentes de conjunto de políticas: 10, 30, 50 y 100.

En cada configuración, se seleccionó aleatoriamente un subconjunto de políticas del conjunto original, y se evaluaron las mismas 2000 solicitudes bajo dos enfoques: *evaluación tradicional*, que examina todas las políticas una por una hasta encontrar coincidencia, y el *modelo propuesto*, que organiza las políticas para reducir el espacio de búsqueda.

Los resultados experimentales, resumidos en la Tabla VI, muestran una mejora consistente del modelo propuesto en términos de tiempo de evaluación y eficiencia de búsqueda. Para todos los valores de N, el modelo propuesto logra un mejor desempeño que el enfoque tradicional.

Además del tiempo, el modelo propuesto examina menos políticas por evaluación. Con N=100, el enfoque tradicional analiza todas, mientras que el propuesto en promedio 24, reduciendo hasta un 76 % el espacio de búsqueda. No se observaron discrepancias en las decisiones de autorización o denegación, lo que valida su

Cuadro VI
COMPARATIVA ENTRE EVALUACIÓN TRADICIONAL Y MODELO PROPUESTO.

Métrica	**N=10**		**N=30**		**N=50**		**N=100**	
	Tradicional	**Propuesta**	**Tradicional**	**Propuesta**	**Tradicional**	**Propuesta**	**Tradicional**	**Propuesta**
Tiempo total [s]	0.100	0.054	0.218	0.085	0.386	0.125	0.676	0.202
Tiempo por solicitud [s]	0.000050	0.000027	0.000109	0.000043	0.000193	0.000062	0.000338	0.000101
Políticas escaneadas	∼10	∼3	∼30	∼8	∼50	∼12	∼100	∼24
Speedup	1.860x		2.550x		3.100x		3.360x	

equivalencia semántica respecto al enfoque tradicional, con mayor eficiencia.

V. CONCLUSIONES

Este trabajo presentó un modelo dinámico de control de acceso basado en eventos, donde las políticas se estructuran como combinaciones de atributos clave (sujeto, recurso, ubicación, tiempo y acción).

- Se desarrolló un mecanismo de organización y evaluación que reduce el espacio de búsqueda de políticas, mejorando la eficiencia sin sacrificar la exactitud.
- La validación mediante un prototipo y un estudio de caso con el dataset KDD Cup 1999 mostró que el modelo mantiene un rendimiento estable y escalable, lo que confirma su aplicabilidad en escenarios de seguridad donde se manejan grandes volúmenes de solicitudes.
- En conjunto, los resultados evidencian que la propuesta no solo es conceptualmente viable, sino también relevante en términos prácticos, al proporcionar un camino para implementar mecanismos de control de acceso más adaptativos frente a contextos cambiantes.

VI. TRABAJO FUTURO

Dado que se han obtenido resultados preliminares prometedores, nos planteamos el siguiente trabajo futuro:

- Evaluar el sistema con solicitudes generadas en tiempo real y comportamientos menos estructurados para analizar su capacidad ante patrones desconocidos.
- Incorporar atributos numéricos y condiciones temporales complejas que enriquezcan la expresividad de las políticas.
- Realizar comparaciones directas con enfoques existentes reportados en el estado del arte.

REFERENCIAS

[1] J. Leng, W. Sha, B. Wang, P. Zheng, C. Zhuang, Q. Liu, T. Wuest, D. Mourtzis, and L. Wang, "Industry 5.0: Prospect and retrospect," *Journal of Manufacturing Systems*, vol. 65, p. 279–295, Oct. 2022.

[2] M. A. Hassan, S. Zardari, M. U. Farooq, M. M. Alansari, and S. A. Nagro, "Systematic analysis of risks in industry 5.0 architecture," *Applied Sciences*, vol. 14, p. 1466, Feb. 2024.

[3] S. khan, T. Mazhar, T. Shahzad, M. Amir khan, A. U. Rehman, and H. Hamam, "Integration of smart grid with industry 5.0: Applications, challenges and solutions," *Measurement: Energy*, vol. 5, p. 100031, Mar. 2025.

[4] M. Alramadhan and K. Sha, "An overview of access control mechanisms for internet of things," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, p. 1–6, IEEE, July 2017.

[5] G. Fragkos, J. Johnson, and E. E. Tsiropoulou, "Dynamic role-based access control policy for smart grid applications: An offline deep reinforcement learning approach," *IEEE Transactions on Human-Machine Systems*, vol. 52, p. 761–773, Aug. 2022.

[6] K. Soni and S. Kumar, "Comparison of rbac and abac security models for private cloud," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, p. 584–587, IEEE, Feb. 2019.

[7] A. Suganthy and V. Prasanna Venkatesan, "An introspective study on dynamic role-centric rbac models," in *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, p. 1–6, IEEE, Mar. 2019.

[8] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, p. 85–88, Feb. 2015.

[9] H. F. Atlam and Y. Yang, "Enhancing healthcare security: A unified rbac and abac risk-aware access control approach," *Future Internet*, vol. 17, p. 262, June 2025.

[10] Y. Zhang, M. Yutaka, M. Sasabe, and S. Kasahara, "Attribute-based access control for smart cities: A smart-contract-driven framework," *IEEE Internet of Things Journal*, vol. 8, p. 6372–6384, Apr. 2021.

[11] M. Cheminod, L. Durante, F. Valenza, and A. Valenzano, "Toward attribute-based access control policy in industrial networked systems," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, p. 1–9, IEEE, June 2018.

[12] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, p. 1–6, IEEE, July 2009.

[13] M. Yang, V. Atluri, S. Sural, and J. Vaidya, *A Graph-Based Framework for ABAC Policy Enforcement and Analysis*, p. 3–23. Springer Nature Switzerland, 2024.

[14] M. Liu, C. Yang, H. Li, and Y. Zhang, "An efficient attribute-based access control (abac) policy retrieval method based on attribute and value levels in multimedia networks," *Sensors*, vol. 20, p. 1741, Mar. 2020.

[15] P. Paul and S. Sural, "Towards efficient evaluation of abac policies using high-dimensional indexing techniques," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, p. 243–251, IEEE, Dec. 2021.

[16] Q. Hu, M. Correia, and T. Jiang, "An efficient blockchain for decentralized abac policy decision point," *Future Generation Computer Systems*, vol. 166, p. 107732, May 2025.

Cifrado homomórfico como servicio para tareas de minería de datos con preservación de privacidad

Shanel Reyes-Palacios
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
shanel.reyes@cinvestav.mx

Miguel Morales-Sandoval
Ciencias Computacionales, INAOE
Tonantzintla, Pue, Mexico
mmorales@inaoep.mx

Jose Juan Garcia-Hernandez
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
jjuan.garcia@cinvestav.mx

J.L. Gonzalez-Compean
Cinvestav Tamaulipas
Ciudad Victoria, Tamps, Mexico
joseluis.gonzalez@cinvestav.mx

Heidy M Marin-Castro
UPIIAP-IPN
Puebla, Pue, Mexico
heidy.marinc@gmail.com

***Abstract*—El cifrado homomórfico es una técnica que ha permitido desarrollar métodos de minería de datos (clasificación, agrupamiento) con preservación de privacidad (MDPP), al realizar operaciones directamente sobre datos cifrados, preservando la confidencialidad de la información. Sin embargo, su uso práctico sigue estando limitado por el elevado costo computacional que implica su ejecución. Este trabajo presenta el diseño y arquitectura de un servicio de cifrado homomórfico, validado con los esquemas de cifrado homomórfico mayormente usados en MDPP: Liu, CKKS y Paillier. El diseño puede soportar cualquier otro esquema de cifrado, incorpora patrones de paralelismo y esquemas de distribución de carga de trabajo. Estas características le permiten al servicio mitigar la sobrecarga asociada al costo computacional que implica la ejecución de los esquemas de cifrado homomórfico. Realizamos una evaluación con distintos tamaños de conjuntos de datos en el contexto de MDPP. Se analizó el rendimiento y la escalabilidad del servicio y se verificó la conservación de la utilidad en tareas de agrupamiento. Los resultados muestran que CKKS logró reducciones de tiempo de hasta el 88% frente a la ejecución secuencial, mientras que Liu y Paillier alcanzaron reducciones entre el 60% y el 70%, sin comprometer la calidad de los datos, lo que posiciona al servicio como una solución factible para MDPP.**

***Index Terms*—Minería de datos como servicio, cifrado homofórfico, escalabilidad, agrupamiento.**

I. Introducción

El cifrado homomórfico (Homomorphic Encryption, por sus siglas en inglés) permite realizar operaciones directamente sobre datos cifrados sin necesidad de descifrarlos, garantizando la privacidad en escenarios como cómputo delegado a terceros [1], aprendizaje automático seguro [2] y minería de datos con preservación de la privacidad (MDPP) [3]. No obstante, a pesar de su potencial, su elevado costo computacional limita su aplicación práctica en entornos de big data, donde los volúmenes de datos y la complejidad de las tareas pueden provocar tiempos de procesamiento inaceptables [4], [5]. Dado que muchas tareas de mineria de datos como clustering y clasificación, requieren preservar la privacidad de los datos, es crucial mejorar los tiempos de cifrado sin sacrificar utilidad. Por ello, en este trabajo validamos nuestro servicio con los tres esquemas más comunes en MDPP: Liu, CKKS y Paillier, cada uno ofreciendo diferentes compromisos entre funcionalidad y costo computacional.

Liu [6] es un esquema simétrico completamente homomórfico que admite suma, multiplicación, resta y multiplicación por escalar con costo lineal en el tamaño de la clave y del texto cifrado. CKKS [7] implementa un cifrado aproximado sobre números reales codificados en polinomios, ideal para aplicaciones de aprendizaje automático tolerantes a ligeras imprecisiones. Paillier [8] es un esquema asimétrico parcialmente homomórfico aditivo, mejorado para operaciones de suma y multiplicación por escalar sobre enteros con textos relativamente compactos.

Proponemos un servicio de cifrado homomórfico, validado con el algoritmo de Liu, CKKS y Paillier pero extensible a otros esquemas. A partir de la configuración del usuario, el servicio segmenta el dataset por filas con un tamaño configurable, asigna los segmentos a procesos según la carga, cifra cada bloque y consolida los resultados junto con sus metadatos. La interfaz abstrae el backend criptográfico, lo que facilita intercambiar o incorporar esquemas. Este flujo reduce los tiempos de cifrado y mantiene la estructura por registro requerida por algoritmos de ciencia de datos con preservación de privacidad, como en aquellos para agrupamiento y clasificación.

La minería de datos con preservación de la privacidad se compone de dos etapas: primero, la preparación de los datos para preservar la privacidad, por ejemplo, usando cifrado homomórfico, y a continuación, la aplicación del algoritmo de minería de datos (clustering o clasificación). Nuestra propuesta aborda esa primera etapa, donde se segmenta el conjunto de datos, se cifra cada segmento de manera paralela y finalmente cada uno se entregan a la segunda etapa sin necesidad de trabajo adicional.

Este servicio ha sido validado mediante una evaluación en la que se define un conjunto de datos de 100,000 registros por 100 atributos, se varía el número de procesos y el esquema de cifrado. Medimos rendimiento y escalabilidad, y verificamos que los resultados cifrados conservan su utilidad en tareas de

agrupamiento. Los resultados muestran reducciones significativas en los tiempos de cifrado sin sacrificar la privacidad, lo que posiciona este servicio como una solución práctica para análisis escalable con preservación de la privacidad.

El resto del artículo se organiza de la siguiente manera. En la Sección II presentamos los conceptos y notación básicos de HE. En la Sección III revisamos los trabajos relacionados con modos de ejecución local y paralelo en HE. La Sección IV describe el diseño e implementación de nuestro servicio. La Sección V detalla el entorno experimental, las métricas y los resultados de rendimiento y utilidad. Finalmente, la Sección VI concluye el artículo y plantea líneas de trabajo futuras.

II. Preliminares

En este trabajo empleamos esquemas de cifrado homomórfico, que permiten realizar operaciones aritméticas directamente sobre datos cifrados sin necesidad de descifrarlos. Para ello, definimos las operaciones homomórficas mediante los operadores:

$$E_3 = E_1 \oplus E_2, \qquad E_3 = E_1 \otimes E_2, \qquad E' = c \odot E_1,$$

donde E_1 y E_2 son textos cifrados, y c es un escalar en claro. Un esquema se describe con los algoritmos

$$CT \leftarrow \mathsf{Enc}\big(K_{\text{enc}}, M\big), \qquad M \leftarrow \mathsf{Dec}\big(K_{\text{dec}}, CT\big),$$

donde M es el mensaje en claro y CT su correspondiente texto cifrado. K_{enc} es la clave de cifrado y K_{dec} la clave de descifrado. Si $K_{\text{enc}} = K_{\text{dec}}$, el esquema es simétrico; en caso contrario, es asimétrico. Siempre debe cumplirse

$$\mathsf{Dec}\big(K_{\text{dec}}, \mathsf{Enc}(K_{\text{enc}}, M)\big) = M$$

Los esquemas de cifrado homomórfico se distinguen por la cantidad y el tipo de operaciones que admiten antes de requerir un descifrado intermedio [9]. Los esquemas parcialmente homomórficos (PHE) soportan solo un tipo de operación, ya sea suma o multiplicación. Los esquemas algo homomórficos (SWHE) extienden esta capacidad permitiendo un número limitado de sumas y productos, mientras que los esquemas totalmente homomórficos (FHE) admiten un número arbitrario de ambas operaciones. En este artículo validamos el servicio con tres esquemas representativos de estas clases: Paillier (PHE aditivo), Liu (FHE simétrico con aritmética entera) y CKKS (FHE aproximado para datos decimales). Cada uno de estos esquemas se define mediante tres algoritmos básicos: 1) generación de llaves (`KeyGen`), 2) cifrado (`Encrypt`), y 3) descifrado (`Decrypt`). A continuación se presenta el pseudocodigo de cada esquema: el Algoritmo 1 resume `KeyGen`, `Encrypt` y `Decrypt` del esquema de Paillier, los Algoritmos 2 y 3 describen los mismos procedimientos para los esquemas de Liu y CKKS respectivamente, incluyendo la codificación/decodificación y el manejo de escala. Las operaciones homomórficas se expresan de forma uniforme con los operadores $\oplus$, $\otimes$ y $\odot$:

a) Paillier (PHE aditivo): Fundamentado en el problema del residuo cuadrático [8], con espacio de mensajes $\mathbb{Z}_n$.

Algorithm 1 Paillier — `KeyGen`, `Encrypt`, `Decrypt`

procedure KeyGen(k)
  Elegir primos p, q de k bits
  $n \leftarrow p \cdot q, \quad \lambda \leftarrow \operatorname{lcm}(p-1, q-1)$
  Elegir $g \in \mathbb{Z}_{n^2}^*$
  $\mu \leftarrow \big(L(g^\lambda \bmod n^2)\big)^{-1} \bmod n$, donde $L(u) = \frac{u-1}{n}$
  Salida: $K_{\text{enc}} = (n, g)$, $K_{\text{dec}} = (\lambda, \mu)$
end procedure
procedure Encrypt(K_{enc}, M)
  Elegir $r \in \mathbb{Z}_n^*$
  $CT \leftarrow g^M \cdot r^n \bmod n^2$
  Salida: CT
end procedure
procedure Decrypt(K_{dec}, CT)
  $M \leftarrow L\big(CT^\lambda \bmod n^2\big) \cdot \mu \bmod n$
  Salida: M
end procedure

Las operaciones homomórficas admitidas son $E_3 = E_1 \otimes E_2 = v_1 \cdot v_2 \bmod n^2$, $E' = c \odot E = E^c \bmod n^2$, (donde v es un plaintext y c es un escalar en claro), lo que permite implementaciones de suma, multiplicación por escalar y resta cifrada.

b) Liu (FHE simétrico): $\mathbb{R}$ cifra un valor $v \in \mathbb{R}$ transformandolo en un vector $E_v \in \mathbb{R}^m$ [6].

Algorithm 2 Liu — `KeyGen`, `Encrypt`, `Decrypt`

procedure KeyGen(m)
  Generar $SK_m = \big[(k_1, s_1, t_1), (k_2, s_2, t_2), \cdots k_m, s_m, t_m)\big]$ con $(k_i, s_i, t_i) \in \mathbb{R}$
  Salida: $K_{\text{enc}} = K_{\text{dec}} = SK_m$
end procedure
procedure Encrypt(K_{enc}, v)
  Generar $r_1, \ldots, r_m \in \mathbb{R}$ aleatorios
  for $i = 1$ **to** $m-1$ **do**
    $e_i \leftarrow k_i t_i v + s_i r_m + k_i (r_i - r_{m-1})$
  end for
  $e_m \leftarrow (k_m + s_m + t_m)\, r_m$
  $CT \leftarrow E_v = [e_1, \ldots, e_m]$
  Salida: CT
end procedure
procedure Decrypt(K_{dec}, CT)
  $t = \sum_{i=1}^{m-1} t_i$
  $s = \dfrac{e_m}{k_m + s_m + t_m}$
  $v = \dfrac{\sum_{i=1}^{m-1} \big(e_i - s \cdot s_i\big)/k_i}{t}$
  Salida: v
end procedure

Las primitivas homomórficas se definen por componente: $E_3 = E_1 \oplus E_2$, $E_3 = E_1 \otimes E_2$, $E' = c \odot E$, $E_3 = E_1 \ominus E_2$, lo que cubre suma, multiplicación, multiplicación por escalar y resta.

Algorithm 3 CKKS — `KeyGen`, `Encrypt`, `Decrypt`

procedure KEYGEN(n, q, σ)
  Generar secreto $K_{\text{dec}}(x)$ (polinomio pequeño) en $\mathbb{Z}_q[x]/(x^n+1)$
  Elegir $a(x)$ uniforme; muestrear ruido $e(x)$ gaussiano
  $b(x) \leftarrow \big(-a(x)\,K_{\text{dec}}(x) + e(x)\big) \bmod q$
  Salida: $K_{\text{enc}} = (a(x), b(x))$, $K_{\text{dec}}(x)$
end procedure
procedure ENCRYPT(K_{enc}, v, scale)
  Codificar $v \mapsto m(x)$ con *scale*
  Muestrear $u(x)$, $e_1(x)$, $e_2(x)$
  $c_1(x) \leftarrow \big(m(x) + b(x)u(x) + e_1(x)\big) \bmod q$
  $c_2(x) \leftarrow \big(a(x)u(x) + e_2(x)\big) \bmod q$
  Salida: $CT = (c_1(x), c_2(x))$
end procedure
procedure DECRYPT(K_{dec}, CT, scale)
  $m^*(x) \leftarrow \big(c_1(x) + c_2(x)\,K_{\text{dec}}(x)\big) \bmod q$
  Decodificar $m^*(x) \mapsto \tilde{v}$ (aproximación de v)
  Salida: $\tilde{v}$
end procedure

c) CKKS (FHE aproximado): Realiza cifrado de numeros reales basado en el problema RLWE [7]. Opera en el anillo $\mathbb{Z}_q[x]/(x^n+1)$ y permite empaquetar vectores de reales (batching). Admite operaciones de suma, multiplicación, rotación y relinearización, lo que lo hace apto para aprendizaje automático con tolerancia a pequeñas imprecisiones.

Cada uno de estos esquemas se asocia a diferentes costos computacionales y tamaños de texto cifrado. Paillier genera textos cifrados compactos, pero solo admite sumas y multiplicaciones por un escalar. En cambio, Liu y CKKS son completamente homomórficos. Liu produce vectores cifrados de tamaño lineal en m y ofrece aritmética exacta, mientras que CKKS empaqueta múltiples valores en un único polinomio de grado n, lo que aumenta el tamaño del cifrado y la complejidad de controlar el ruido a cambio de permitir operaciones aproximadas sobre lotes de gran volumen.

Aunque el foco de este artículo son los esquemas de cifrado, estos pueden emplearse para soportar tareas de minería de datos con preservación de la privacidad, como clustering y clasificación [10]–[12].

III. Estado del arte

Se llevó a cabo un estudio del estado del arte centrado en trabajos publicados en los últimos cinco años, que abordan cifrado homomórfico aplicado a tareas de minería de datos con preservación de la privacidad (MDPP), enfocado en modos de ejecución *local* y *paralelo* en un único nodo. Se priorizaron propuestas con evaluación experimental y una descripción clara del modo de ejecución, y se excluyeron soluciones que no emplean HE. La Tabla I resume las propuestas seleccionadas y su clasificación por modo de ejecución.

En el modo de ejecución local, tanto la tarea de cifrado como la tarea de minería de datos se ejecutan en una única máquina sin recurrir a hilos ni procesos adicionales. Yunlu Cai *et al.* [10] presentan un protocolo de dos partes que ejecuta k-means homomórfico en un único servidor, operando sobre datos cifrados con el esquema de Liu y sin paralelización. Lekshmy *et al.* [11] proponen un enfoque híbrido para acelerar k-means cifrado con Paillier mediante mini-batches, reduciendo el número de operaciones homomórficas por iteración en una sola instancia de cómputo; sin embargo, la tarea de cifrado se mantiene de manera secuencial. Rovida *et al.* [17] estudian una versión aproximada de k-means basada en CKKS y técnicas de enmascaramiento, en la que el servidor ejecuta la mayoría de las operaciones de agrupamiento cifrado y el cliente solo aplica una máscara al resultado, lo que mejora el rendimiento sin emplear paralelismo explícito. Sokhonn *et al.* [12] describen un método cooperativo de agrupamiento jerárquico sobre CKKS, en el cual el propietario de los datos asiste al servidor durante el ordenamiento de distancias cifradas para completar el proceso en una única máquina. El esquema propuesto por Jiasen *et al.* [15] implementa k-nn seguro sobre datos cifrados con CKKS en un unico servidor en la nube. Asimismo, Zorarpacı *et al.* [16] presentan una estrategia híbrida que combina el cifrado aditivo de Paillier con técnicas de privacidad diferencial, aplicada a clasificadores como One Rule y Naïve Bayes, en este caso, tanto el cifrado como la aplicación de ruido se realizan en un único entorno.

Tabla I
TRABAJOS RELACIONADOS CON EL MODO DE EJECUCIÓN LOCAL Y PARALELO PARA CIFRADO HOMOMÓRFICO.

Autor	Año	Esquema HE	Modo de ejecución
Yunlu Cai *et al.* [10]	2019	Liu	Local
Almutairi *et al.* [13]	2020	Liu	Paralelo
Lekshmy *et al.* [11]	2020	Paillier	Local
Xuancheng Guo *et al.* [14]	2020	Paillier	Paralelo
Jiasen *et al.* [15]	2021	CKKS	Local
Zorarpacı *et al.* [16]	2022	Paillier	Local
Rovida *et al.* [17]	2023	CKKS	Local
Sokhonn *et al.* [12]	2024	CKKS	Local

En el modo de ejecución paralelo se emplean múltiples hilos o procesos dentro de la misma máquina o servicio para distribuir la carga de trabajo. Almutairi *et. al* [13] diseñan un servicio DMaaS en la nube que orquesta contenedores encargados de procesar una matriz de distancia global con el esquema de Liu y MUOPE, escalando dinámicamente el número de procesos de ejecución según la demanda. De manera similar, Xuancheng Guo *et. al* [14] implementan una plataforma MLaaS basada en microservicios y un bus MQTT, donde varios procesos colaboran en el clustering homomórfico utilizando Paillier y RSA para mejorar la latencia extremo a extremo y la sobrecarga de comunicación. Estas propuestas muestran beneficios del paralelismo en un nodo, pero no exponen una interfaz unificada independiente del esquema HE ni detallan una segmentación por filas con balanceo de carga explícito en la fase de *cifrado* previa a la MDPP.

A. Discusión

Los trabajos revisados cubren adecuadamente los modos local y paralelo, pero ninguno ofrece un servicio de cifrado homomórfico agnóstico al esquema que segmente el dataset y distribuya los segmentos de manera balanceada entre los procesos disponibles. Nuestra propuesta aborda este desafío mediante la propuesta de un servicio que aplica patrones de

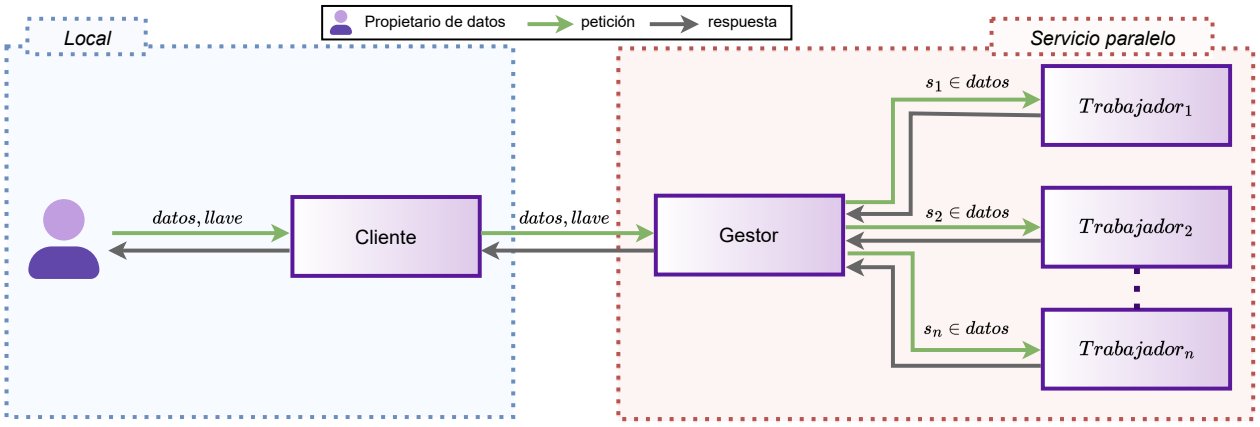


Figura 1. Arquitectura conceptual del servicio paralelo.

paralelismo y balanceo de carga, asignando a cada proceso segmentos proporcionales a su capacidad. Este servicio expone una interfaz única desacoplada del backend criptográfico. El flujo se organiza con el patrón gestor–trabajador para cifrar en paralelo dentro de un único nodo y se valida con tres esquemas representativos (Liu, CKKS y Paillier), facilitando su integración en flujos de MDPP.

IV. Cifrado homomórfico como servicio

El servicio de cifrado homomórfico ofrece una interfaz unificada para distintos esquemas HE. Está diseñado para operar de manera paralela, por lo que la carga de trabajo se divide entre múltiples procesos en el mismo nodo, aprovechando al máximo los recursos de CPU disponibles para reducir los tiempos de cifrado.

El servicio divide en segmentos el conjunto de datos a cifrar, lanza los procesos de cifrado para cada segmento en paralelo, recolectando los resultados de forma transparente.

Consideramos un dataset $D \in \mathbb{R}^{n\times a}$, donde cada fila representa un registro con a atributos. El cifrado conserva la estructura por registro, generando una matriz D', de manera que cada elemento $r_{i,j} \in D$ se transforma en un texto cifrado $E_{i,j} \in D'$:

$$D = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ r_{k,1} & r_{k,2} & \cdots & r_{k,a} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,a} \end{bmatrix} \xrightarrow{\text{Encrypt}} D' = \begin{bmatrix} E_{1,1} & E_{1,2} & \cdots & E_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ E_{k,1} & E_{k,2} & \cdots & E_{k,a} \\ \vdots & \vdots & \ddots & \vdots \\ E_{n,1} & E_{n,2} & \cdots & E_{n,a} \end{bmatrix}$$

Para aprovechar el paralelismo, se segmenta por filas usando un tamaño de segmento TS y el número de segmentos $S = \lceil n/TS \rceil$. Definimos el dataset segmentado como:

$$D = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \vdots \\ d^{(s)} \end{bmatrix}$$

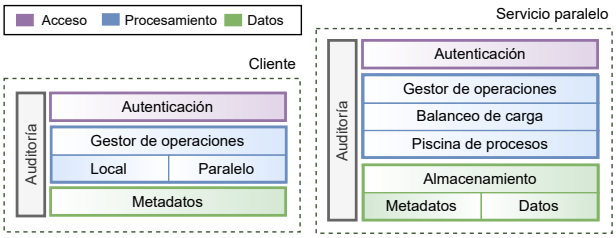


Figura 2. Arquitectura en capas de los componentes del servicio

se realiza el proceso en paralelo

$$d'^{(j)} = \texttt{Encrypt}(K_{enc},\, d^{(j)})$$

y se integra para entregar un resultado final al propietario de datos:

$$D' = \left[d'^{(1)};\, d'^{(2)};\, \ldots;\, d'^{(s)}\right]$$

Este diseño modular, centrado en patrones de paralelismo y balanceo, facilita la integración de cifrado homomórfico en flujos de análisis de datos, mejorando la privacidad sin sacrificar el rendimiento.

La Figura 1 ilustra de manera conceptual el flujo de interacción con el servicio. El proceso inicia del lado del propietario de los datos, que dispone de un cliente. Este cliente facilita la comunicación con el servicio, permitiendo configurar distintos parámetros (cantidad de segmentos y procesos disponibles). Además, ofrece operaciones básicas como generación de llaves, cifrado, descifrado y segmentación. La petición enviada de parte del cliente es recibida por un *gestor* que segmenta el conjunto de datos y lanza múltiples procesos de cifrado. Cada proceso cifra su correspondiente segmento y devuelve el resultado al *gestor*, que agrupa las salidas y entrega al *cliente* los datos cifrados junto con los metadatos asociados. Esta separación permite adaptar el flujo interno para mejorar la eficiencia del cifrado homomórfico.

La Figura 2 presenta la solución organizada en tres capas para dos componentes principales: cliente y servicio paralelo. En ambos componentes, la capa de acceso valida credenciales y parámetros de ejecución antes de aceptar una petición.

En el componente *cliente*, la capa de procesamiento maneja el flujo según la configuración indicada por el usuario: selecciona el esquema de cifrado, determina el tamaño de segmento y el número de procesos, aplica la segmentación por filas y prepara las tareas

para su ejecución en el nodo. Para fines de comparación, es posible ejecutar en modo local, reutilizando el mismo flujo, o bien en modo paralelo, coordinando el envío de segmentos al *servicio paralelo* y la integración en el orden original al finalizar. La capa de datos del componente cliente administra los metadatos de la solicitud y de cada segmento.

En el componente *servicio paralelo*, la capa de procesamiento recibe los segmentos y los distribuye a un pool de procesos en el mismo nodo aplicando balanceo de carga. Cada proceso ejecuta el cifrado del segmento con el esquema indicado, aplica políticas de reintento ante fallos o tiempo excedido, y devuelve resultados parciales al gestor para su agregación, preservando el orden lógico de los segmentos y la integración transparente con el cliente. La capa de datos gestiona los metadatos de ejecución a nivel de proceso y segmento (estado, número de reintentos, tiempos parciales).

De manera transversal, un módulo de auditoría registra eventos y métricas (tiempos de respuesta, uso de CPU/RAM, reintentos y estado) para asegurar trazabilidad y diagnóstico durante toda la ejecución.

V. Evaluación

Con el objetivo de analizar la eficiencia del servicio de cifrado homomórfico propuesto, realizamos tres experimentos que miden el tiempo de respuesta en función del algoritmo y del nivel de seguridad. En todos los casos se emplearon los siguientes backends criptográficos: CKKS con Pyfhel, Paillier con la librería phe y Liu con una implementación propia. El servicio es agnóstico al backend, pues el motor criptográfico se desacopla mediante interfaces; por ello, la evaluación se centra en la ganancia del paralelismo frente a una version secuencial. Primero, evaluamos la escalabilidad con el número de procesos en tres niveles de seguridad (Fig. 3–5), después, comparamos los modos local y paralelo con 16 procesos fijos (Fig.6), por último, verificamos la utilidad comparando el agrupamiento cifrado frente al agrupamiento estándar (Fig. 7).

A. Evaluación 1: Escalabilidad con número de procesos de cifrado

En esta evaluación se midió el tiempo de respuesta del proceso de cifrado empleando los esquemas Liu, Paillier y CKKS en tres niveles de seguridad: 128, 192 y 256 bits. En cada caso, se procesó un conjunto de datos de 100,000 registros por 100 atributos y se varió el número de procesos en cinco configuraciones (1, 2, 4, 8 y 16).

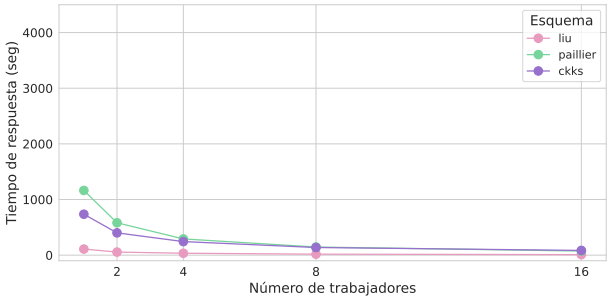


Figura 3. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 128 bits.

La Figura 3 presenta los resultados para 128 bits de seguridad, donde se observa una reducción notable del tiempo de cifrado al incrementar los procesos disponibles. Liu y Paillier alcanzan tiempos de respuesta de 10 y 73 segundos con 16 procesos, mientras que CKKS, más costoso, cae de 735 segundos a 85 segundos.

En la Figura 4, correspondiente a 192 bits, se mantiene la tendencia de mejora, con tiempos máximos iniciales mayores (hasta 2100 segundos para el caso de Paillier) pero un escalado proporcional similar.

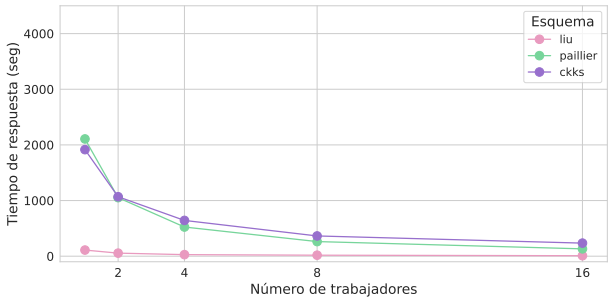


Figura 4. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 192 bits.

Finalmente, la Figura 5 muestra que para 256 bits, el paralelismo sigue siendo efectivo: aunque los tiempos para un proceso aumentan (4207 segundos para CKKS), al usar 16 procesos todos los esquemas reducen sus tiempos de manera significativa (Liu 9 segundos, Paillier 251 segundos y CKKS 509 segundos), confirmando el impacto positivo del balanceo de carga.

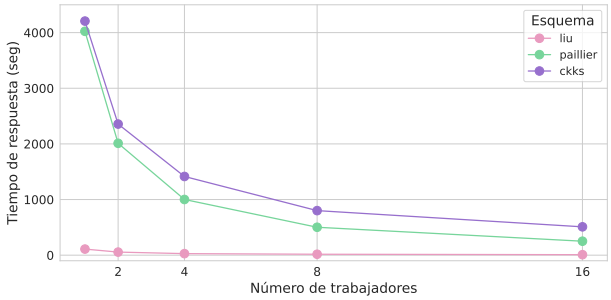


Figura 5. Tiempo de respuesta para distintos esquemas de cifrado al variar la cantidad de procesos con un nivel de seguridad de 256 bits.

Estos resultados validan que el servicio aprovecha eficientemente los recursos computacionales en modo paralelo, atenuando la sobrecarga de cifrado homomórfico incluso en niveles de seguridad mayores.

B. Evaluación 2: Comparativa de modos de ejecución

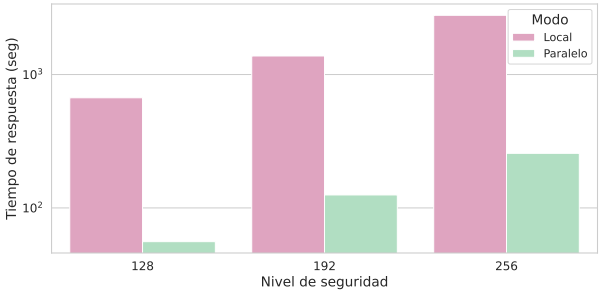


Figura 6. Comparativa de tiempo de respuesta entre modos de ejecución para diferentes niveles de seguridad.

En esta evaluación se comparan los dos modos de ejecución: local y paralelo. Para esta prueba, en el modo paralelo se mantuvo constante el número de procesos (dieciséis). Los tiempos de respuesta se presentan en la Figura 6. En todos los casos, el modo paralelo reduce el tiempo de cifrado de manera sustancial frente al modo local.

C. Evaluación 3: Validación de utilidad

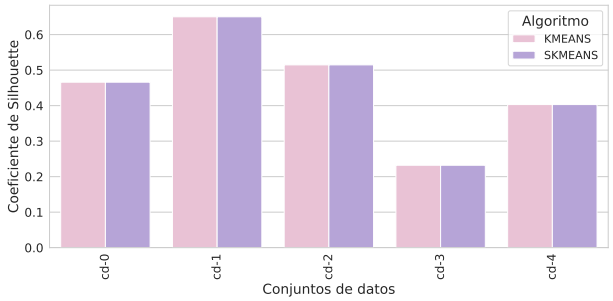


Figura 7. Comparativa de coeficiente de silhouette para los algoritmos k-means y sk-means con el esquema de Liu.

Para comprobar que el cifrado homomórfico no degrada la calidad de los resultados en minería de datos, realizamos una prueba de concepto sobre cinco conjuntos del repositorio UCI (Ecoli, Iris, Breast Cancer, Audit data y Fertility Diagnosis). Se aplicaron los algoritmos kmeans para texto en claro y skmeans para textos cifrados, para este segundo caso, cada conjunto de datos fue segmentado y cifrado usando el esquema de Liu con 128 bits, (en modo paralelo) o cifrado de forma completa (modo local). La Figura 7 muestra los coeficientes de silhouette obtenidos por ambos métodos, con valores que oscilan entre 0.60 y 0.75. En todas las pruebas, k-means y sk-means alcanzan valores idénticos de silhouette , lo que confirma que la fase de cifrado, incluso segmentada y paralelizada, no introduce pérdida de utilidad y preserva la misma calidad de agrupamiento que el algoritmo sin cifrado

VI. Conclusión

En este trabajo se presentó un servicio de cifrado homomórfico, validado con los esquemas Liu, CKKS y Paillier, que ofrece una interfaz unificada respaldada por una arquitectura modular basada en patrones de paralelismo y balanceo de carga. Las pruebas realizadas muestran que aplicar el modo paralelo reduce significativamente los tiempos de cifrado frente a una ejecución secuencial, logrando mejoras de hasta un 75 % en CKKS y entre un 60 % y un 70 % en Liu y Paillier.

Estos resultados confirman que segmentar y distribuir las tareas de cifrado entre procesos mejora el uso de los recursos dentro de un único nodo, mitigando así su elevado costo computacional. Además, la prueba de utilidad con k-means y sk-means demostró que la versión cifrada conserva idénticos coeficientes de silhouette en cinco conjuntos de datos, validando que no hay pérdida de calidad en el agrupamiento.

En conjunto, el servicio facilita la incorporación del cifrado homomórfico en flujos de análisis de datos con preservación de la privacidad, combinando rendimiento y facilidad de uso. Como trabajo futuro, contemplamos: 1) su extensión a entornos distribuidos multinodo, manteniendo la misma interfaz y el patrón de segmentación y ejecución en paralelo, y 2) la integración y evaluación de aceleración por GPU en los núcleos más costosos (por ejemplo, multiplicaciones y rotaciones polinomiales en CKKS, operaciones vectorizadas en Liu y exponenciaciones modulares en Paillier) con el fin de cuantificar su impacto en tiempos de cifrado.

References

[1] M. Ileas Pramanik, Raymond Y.K. Lau, Md. Sakir Hossain, Md. Mizanur Rahoman, Sumon Kumar Debnath, Md. Golam Rashed, and Md. Zasim Uddin. Privacy-preserving big data analytics: A critical analysis of state-of-the-art. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1387, 2021.

[2] Kathrin Grosse, Lukas Bieringer, Tarek R. Besold, Battista Biggio, and Katharina Krombholz. Machine learning security in industry: A quantitative survey. *IEEE Transactions on Information Forensics and Security*, 18:1749–1762, 2023.

[3] A. Asaduzzaman and ... D'Souza. Increase security by analyzing password strength using machine learning. In *Proceedings of the 2024 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTIDAMT & NCON)*. IEEE, 2024.

[4] Bhomik M. Gandhi, Shruti B. Vaghadia, Malaram Kumhar, Rajesh Gupta, Nilesh Kumar Jadav, Jitendra Bhatia, Sudeep Tanwar, and Abdulatif Alabdulatif. Homomorphic encryption and collaborative machine learning for secure healthcare analytics. *Security & Privacy*, 8(1), 2024. Published September 13, 2024.

[5] Fabian Schmid, Shibam Mukherjee, Stjepan Picek, Marc Stöttinger, Fabrizio De Santis, and Christian Rechberger. Towards private deep learning-based side-channel analysis using homomorphic encryption: Opportunities and limitations. In *Constructive Side-Channel Analysis and Secure Design (COSADE 2024)*, volume 14595 of *Lecture Notes in Computer Science*, pages 133–154, 2024.

[6] Dongxi Liu. Practical fully homomorphic encryption without noise reduction. Cryptology ePrint Archive, Paper 2015/468, 2015. https://eprint.iacr.org/2015/468.

[7] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. Approximate homomorphic encryption with reduced approximation error. In *Cryptographers' Track at the RSA Conference*, pages 120–144. Springer, 2022.

[8] Sudhansu Bala Das, Sugyan Kumar Mishra, and Anup Kumar Sahu. A new modified version of standard rsa cryptography algorithm. In *Smart Computing Paradigms: New Progresses and Challenges: Proceedings of ICACNI 2018, Volume 2*, pages 281–287. Springer, 2020.

[9] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), jul 2018.

[10] Yunlu Cai and Chunming Tang. Privacy of outsourced two-party k-means clustering. *Concurrency and Computation: Practice and Experience*, 33(8):e5473, 2021.

[11] PL Lekshmy and M Abdul Rahiman. Hybrid approach to speed-up the privacy preserving kernel k-means clustering and its application in social distributed environment. *Journal of Network and Systems Management*, 28(2):398–422, 2020.

[12] Lynin Sokhonn, Yun-Soo Park, and Mun-Kyu Lee. Hierarchical clustering via single and complete linkage using fully homomorphic encryption. *Sensors*, 24(15):4826, 2024.

[13] Nawal Almutairi, Frans Coenen, and Keith Dures. A cryptographic ensemble for secure third party data analysis: collaborative data clustering without data owner participation. *Data & Knowledge Engineering*, 126:101734, 2020.

[14] Xuancheng Guo, Hui Lin, Yulei Wu, and Min Peng. A new data clustering strategy for enhancing mutual privacy in healthcare iot systems. *Future Generation Computer Systems*, 113:407–417, 2020.

[15] Jiasen Liu, Chao Wang, Zheng Tu, Xu An Wang, Chuan Lin, and Zhihu Li. Secure knn classification scheme based on homomorphic encryption for cyberspace. *Security and communication networks*, 2021(1):8759922, 2021.

[16] Ezgi Zorarpacı and Selma Ayşe Özel. A hybrid approach of homomorphic encryption and differential privacy for privacy preserving classification. *International Journal of Applied Mathematics Electronics and Computers*, 8(4):138–147, 2020.

[17] Lorenzo Rovida. Fast but approximate homomorphic k-means based on masking technique. *International Journal of Information Security*, 22(6):1605–1619, 2023.

Implementación de un clúster basado en Hadoop y Spark para el entrenamiento de modelos de Machine Learning para la detección de ataques DDoS

José de Jesús Zapata Lara
Instituto Nacional de Astrofísica, Óptica y Electrónica
Tonantzintla, Puebla, México
josedj.zapata@inaoe.mx

***Resumen*—El uso de sistemas distribuidos en los últimos años se ha incrementado debido a las capacidades que ofrecen para el procesamiento y almacenamiento masivo de información, así como por los requerimientos no funcionales que proveen. En el presente trabajo se plantea la implementación de un clúster para el entrenamiento de modelos basados en Machine Learning (ML) para la detección de ataques de negación de servicio distribuidos (DDoS) utilizando el sistema de almacenamiento de archivos distribuido HDFS del framework Hadoop y la integración del framework Spark para procesamiento. Los resultados obtenidos demuestran que un clúster de alto rendimiento que utiliza la integración de Hadoop y Spark puede ser utilizado para el entrenamiento de modelos basados en ML en el contexto de ciberseguridad con buen desempeño en cuanto a precisión, tiempo de ejecución, escalabilidad horizontal y tolerancia a fallos.**

***Palabras clave*—Sistema distribuido, Clúster, Hadoop, Spark, Ataques DDoS, Machine Learning**

I. Introducción

El incremento del uso de sistemas distribuidos en los últimos años está ampliamente relacionado con las capacidades que estos sistemas ofrecen para el procesamiento y almacenamiento masivo de datos [3]. Para la implementación y despliegue de sistemas distribuidos existen modelos de programación y diseño que pueden ser utilizados para atender necesidades específicas de un entorno determinado [11]. Uno de los principales desafíos en el área de ciberseguridad es la detección oportuna de ataques cibernéticos de naturaleza dinámica, como los ataques de negación de servicio distribuidos (DDoS por sus siglas en inglés). Por lo anterior, la tendencia en los últimos años ha sido utilizar modelos basados en Machine Learning (ML) los cuales han demostrado tener la capacidad para detectar patrones de comportamiento anómalo eficientemente sin ser programados de forma explícita [1]. Una de las principales áreas de oportunidad para los modelos basados en ML es realizar su evaluación en escenarios que puedan ser escalables [8]. El insumo principal para los esquemas de detección de ataques DDoS basados en modelos de ML son los conjuntos de datos, los cuales están integrados por archivos que pueden incrementar su volumen rápidamente, por lo cual es importante para estos esquemas utilizar infraestructura de almacenamiento y procesamiento con requerimientos no funcionales como escalabilidad y tolerancia a fallos.

En el estado del arte existen trabajos relacionados con la detección de ataques DDoS mediante cómputo distribuido y modelos basados en ML sin embargo, han sido implementados solo en un equipo de cómputo local, otros han utilizado conjuntos de datos no especializados en ataques DDoS o solamente han sido evaluados para la detección de un ataque DDoS en específico y sin utilizar para la fase de entrenamiento de los modelos las características más relevantes de los ataques DDoS para optimizar la precisión en la detección y para la reducción del tiempo de entrenamiento.

El trabajo realizado plantea una alternativa para almacenar y procesar conjuntos de datos para la fase de entrenamiento de modelos basados en ML para la detección de ataques DDoS en un entorno distribuido con escalamiento horizontal, utilizando las características más relevantes que modelan un ataque DDoS.

Las principales contribuciones del artículo son las siguientes:

- Implementación de un clúster para almacenamiento y procesamiento de las fases de entrenamiento y validación de modelos basados en ML para la detección de ataques DDoS con escalabilidad horizontal y tolerancia a fallos.
- Validación del desempeño de tres modelos basados en ML (Regresión Logística, Random Forest y Gradient Boosted Tree) en un entorno distribuido para la clasificación binaria de ataques DDoS que utilizan los vectores de ataque de Reflexión (DNS y NTP) e Inundación (TCP-SYN y UDP), utilizando las características más relevantes del conjunto de datos.

El resto del artículo está organizado de la siguiente manera: en la Sección II se describen trabajos relacionados con el uso de sistemas distribuidos basados en Hadoop y Spark para la detección de ataques DDoS. La Sección III describe los preliminares del trabajo realizado. La Sección IV describe la metodología utilizada para la implementación y la experimentación. En la sección V se presentan los resultados obtenidos y en la sección VI se presenta la discusión sobre las capacidades de evolución del trabajo realizado, así como sus limitaciones. En la sección VII se presentan las conclusiones y finalmente,

en la sección VIII se presenta la perspectiva de trabajo futuro a partir de las áreas de oportunidad identificadas.

II. Trabajos relacionados

Mediante una revisión sistemática de la literatura se consultaron trabajos relacionados con el uso de Hadoop y Spark para la detección de ataques DDoS. La propuesta del artículo [5] presenta una solución para la detección de ataques DDoS de tipo inundación analizando bitácoras de tráfico de red en tiempo real, lo cual representa una limitante para la detección de ataques DDoS de multiples vectores.

En el trabajo [7] presentan una comparativa del desempeño de los frameworks Hadoop-Mahout y Spark como componente de un sistema para la detección de instrusiones (IDS por sus siglas en inglés), resultando el escenario de Spark con mejor desempeño. La solución propuesta fue entrenada con el conjunto de datos Network Security Lab Knowledge Discovery in Databases (NSL-KDD) el cual incluye características de ataques del tipo Probe, User to Root (U2R), Remote to Local (R2L) y Denial of Service (DoS) y se desarrolló utilizando un entorno de simulación en un equipo de cómputo portátil.

El artículo [6] presenta la propuesta de un modelo para la detección de intrusiones monitoreando el tráfico de red para el entrenamiento de varios modelos basados en ML. En la propuesta se plantea la combinación de multiples fuentes de información y el uso de multiples modelos basados en ML para mejorar el desempeño de un mecanismo de detección de intrusiones, sin embargo no se muestran resultados de la experimentación ni la información sobre el escenario donde fue realizada la implementación.

En el trabajo [12] se presenta una solución basada en Spark implementada en un entorno local para la detección de intrusiones, realizando una comparativa de su desempeño utilizando un conjunto de datos generado por los autores y algunos de los conjuntos de datos más utilizados en el estado del arte, demostrando que la solución desarrollada presenta mejores resultados utilizando el conjunto de datos autogenerado, dado que es más diverso, está actualizado e incluye un mayor número de características.

En el artículo [4] se plantea el despliegue de un escenario distribuido para la clasificación de datos mediante el modelo basado en ML de máquinas de soporte vectorial. En la solución propuesta se plantea la ejecución de procesamiento paralelo utilizando Spark y el almacenamiento de los datos en el sistema de archivos HDFS de Hadoop. El uso de Spark para la detección de ataques DDoS también se plantea de forma similar al trabajo previo en el artículo [9], pero utilizando el modelo Random Forest.

III. Preliminares

III-A. Sistema distribuido tipo clúster

Un sistema distribuido se define como aquel en el que los componentes de hardware y/o software ubicados en equipos de cómputo interconectados mediante una red se comunican y coordinan sus acciones únicamente mediante el envío de mensajes [2]. Los componentes de software fundamentales para el trabajo realizado son Hadoop y Spark. Hadoop es un marco de código abierto que se utiliza para almacenar y procesar de manera eficiente conjuntos de datos grandes de forma masiva y paralela. Spark es un sistema de código abierto para procesamiento distribuido de cargas de trabajo de gran tamaño optimizado mediante el uso de estructuras para almacenamiento de datos en memoria.

III-B. Ataques DDoS

Un ataque DDoS es aquel que tiene como objetivo interrumpir el funcionamiento normal de un sistema o servicio, saturándolo con tráfico de red enviado desde múltiples fuentes, haciéndolo inaccesible para los usuarios legítimos. Uno de los principales desafíos en la detección de los ataques DDoS es su naturaleza dinámica, por lo cual un esquema de detección tradicional ha dejado de ser efectivo en la actualidad.

III-C. Detección de ataques DDoS utilizando ML

Los esquemas de detección de ataques DDoS basados en modelos de Machine Learning (ML) han demostrado ser una alternativa efectiva con altos niveles de precisión y rendimiento para la detección de ataques DDoS [1]. Los modelos basados en ML son eficientes para detectar patrones mediante algoritmos de entrenamiento, lo cual los hace versátiles y con capacidades de generalización.

III-D. Arquitectura del clúster

La arquitectura del clúster implementado es del tipo de alto rendimiento y utiliza nodos de cómputo configurados bajo el modelo de memoria distribuida. El clúster está integrado por un nodo maestro el cual administra la operación y asignación de tareas y por cinco nodos trabajadores que realizan las actividades de procesamiento y almacenamiento. La conectividad de los nodos del clúster es provista por una red con tecnología Gigabit Ethernet configurada con direccionamiento IP privado. Cada nodo del clúster tiene almacenamiento independiente administrado por el sistema de archivos distribuido HDFS del framework Hadoop. La administración de recursos y calendarización de trabajos en el clúster se realiza mediante el componente YARN del framework Hadoop y la ejecución de tareas de forma distribuida con procesamiento de datos en memoria es realizada y administrada mediante el framework Spark.

IV. Metodología

Esta sección describe la metodología utilizada para el desarrollo de la propuesta, la cual estuvo integrada por las etapas de: revisión sistemática de la literatura, la selección y configuración de los elementos de hardware y software, así como la definición del escenario de pruebas para validar su funcionamiento. A continuación se describe el trabajo realizado en cada una de las etapas.

IV-A. Revisión sistemática de la literatura

La revisión realizada consideró artículos de conferencia publicados en las bases de datos: Scopus, IEEE Xplore y SpringerLink. Esta revisión se desarrolló con la finalidad de identificar trabajos relacionados y considerar aquellos que pudieran servir de base.

Las preguntas de investigación utilizadas fueron las siguientes:

1. ¿Un sistema de cómputo distribuido tipo clúster puede ser utilizado para el despliegue eficiente de un escenario para el procesamiento de conjuntos de datos de gran volumen?
2. ¿Existen implementaciones en el estado del arte que utilicen Hadoop y Spark para almacenar y procesar datos en aplicaciones que requieren baja latencia?

Los términos "Hadoop" y "Spark" fueron utilizados en la cadena de búsqueda y la selección de los trabajos se realizó con base en los siguientes criterios de inclusión y exclusión:

- **Criterios de inclusión:** Artículos que utilicen la integración de Hadoop y Spark en el contexto de ciberseguridad y que hayan sido publicados dentro del periodo de 2023 a 2025
- **Criterio de exclusión:** Que el artículo no corresponda a los criterios de inclusión.

El resumen del análisis de la búsqueda se muestra en la tabla I.

Tabla I
ANÁLISIS DE LA BÚSQUEDA

Número de trabajos			
Fuente	***Encontrados***	***Analizados***	***Relacionados al proyecto***
Scopus	57	6	5
IEEE Xplore	21	4	1
SpringerLink	61	8	3

IV-B. Selección y configuración de elementos del clúster

En esta sección se describen los elementos de hardware, software y servicios utilizados. El diagrama del clúster se muestra en la figura 1.

Las especificaciones técnicas de los equipos de cómputo utilizados se muestran en la tabla II.

Para los nodos de cómputo del clúster se utilizó el sistema operativo Linux Ubuntu Server 22.04.5 LTS y fueron interconectados mediante una red privada con el segmento 192.168.2.0/24. Se habilitó el acceso SSH entre los nodos de cómputo mediante llaves para la autenticación sin contraseña y se realizó la instalación y configuración de Hadoop 3.4.1 y de Spark 3.5.6, ambos en modo clúster.

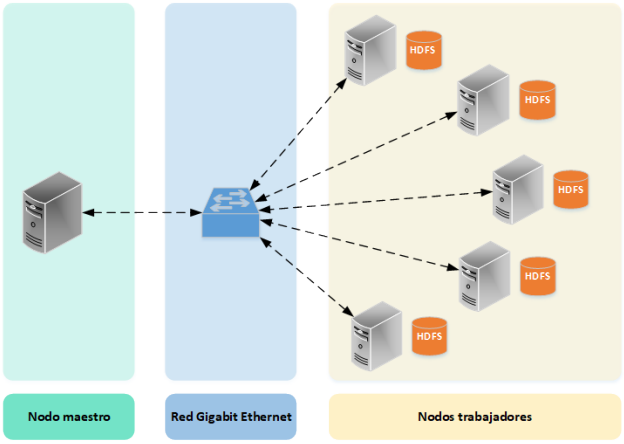


Figura 1. Diagrama de la implementación del clúster.

Tabla II
HARDWARE CLÚSTER HADOOP-SPARK

Especificaciones técnicas			
Descripción	***Procesador***	***Memoria***	***Disco***
Nodo maestro	Intel Xeon X5550 (4 cores)	12 GB	1 TB
Nodos trabajadores	Intel Xeon X5550 (8 cores)	16 GB	1 TB
Cantidades totales	**44 cores**	**92 GB**	**6 TB**

IV-C. Escenario de pruebas

El conjunto de datos utilizado fue CICDDoS-2019 [10], el cual está integrado por capturas de tráfico de ataques DDoS con 80 características y 15 etiquetas. Del conjunto de datos seleccionado se utilizaron los ataques DDoS con los vectores de ataque de reflexión (DNS y NTP) e inundación (TCP SYN y UDP Flood). Los modelos basados en ML seleccionados para la clasificación binaria de ataques DDoS fueron: Regresión Logística (LR), Random Forest (RF) y Gradient Boosted Tree (GBT), con una distribución del 80 % del conjunto de datos para el entrenamiento y el 20 % restante para la validación. Las características utilizadas para el entrenamiento de cada modelo seleccionado se muestran en la tabla III.

Para la evaluación del desempeño de los modelos basados en ML se utilizaron las mediciones Accuracy, Precision, Recall y F1-Score. La descripción de las métricas utilizadas se presenta en la tabla IV.

V. RESULTADOS

La detección de ataques DDoS fue realizada utilizando tres escenarios: el primero mediante el uso de un solo clasificador (LR), el segundo utilizando un ensamble de clasificadores tipo Bagging (RF) y el tercero mediante un ensamble de clasificadores de tipo Boosting (GBT). Se ha demostrado que el uso de un ensamble de clasificadores presenta mejor desempeño que el uso de un solo clasificador y mayor capacidad de generalización en la detección de ataques DDoS [8]. La comparativa de los modelos seleccionados se realizó para

Tabla III
ATAQUES DDOS Y SUS CARACTERÍSTICAS

Ataque DDoS	***Características***
Reflexión DNS	Longitud máxima del paquete, longitud máxima del paquete hacia el destino, longitud mínima del paquete hacia el destino, tamaño promedio de paquete y longitud mínima del paquete.
Reflexión NTP	Promedio de bytes en un subflujo hacia el origen, longitud total de los paquetes hacia el destino, desviación estándar de la longitud del paquete hacia el destino, tamaño mínimo del segmento hacia el destino y tiempo mínimo entre dos paquetes consecutivos en un flujo.
Inundación TCP-SYN	Número de paquetes con el indicador ACK establecido en el encabezado TCP, tamaño inicial de la ventana TCP hacia el destino en bytes, tamaño mínimo del segmento hacia el destino, tiempo total entre dos paquetes enviados hacia e destino y secuencia de paquetes entre origen y destino.
Inundación UDP	Secuencia de paquetes entre origen y destino, desviación estándar de la longitud del paquete hacia el destino desviación estándar de la longitud del paquete, tamañ mínimo del segmento hacia el destino y el identificado del protocolo.

Tabla IV
MEDICIÓN DEL DESEMPEÑO DE MODELOS BASADOS EN ML

Medición	***Descripción***	***Fórmula***
Accuracy	Proporción de todas las clasificaciones que fueron correctas, ya sean positivas o negativas	$\frac{TN+TP}{TN+TP+FN+FP}$
Precision	Proporción de todas las clasificaciones positivas del modelo que son realmente positivas	$\frac{TP}{TP+FP}$
Recall	Proporción de todos los positivos reales que se clasificaron correctamente como positivos	$\frac{TP}{TP+FN}$
F1-Score	Media armónica de Precision y Recall, utilizada para medir con mayor precisión la exactitud del modelo en conjuntos de datos desequilibrados	$2 * \frac{Precision*Recall}{Precision+Recall}$

validar su desempeño en un entorno distribuido y de esta manera explorar su viabilidad de integración en una solución escalable. En los resultados obtenidos se pudo observar que los modelos basados en el ensamble Random Forest y Gradient Boosted Tree presentan un mejor rendimiento en general en comparativa con el modelo de Regresión Logística, debido a que están integrados por múltiples clasificadores que retroalimentan al modelo y de esta manera obtienen mejores resultados en sus predicciones. Al comparar los dos modelos de tipo ensamble fue posible identificar que Random Forest es más rápido que Gradient Boosted Tree, ya que utiliza de forma paralela múltiples clasificadores mediante árboles de decisión, los cuales realizan su entrenamiento sobre muestras aleatorias del conjunto de datos (subconjuntos), obteniendo su predicción final a partir del promedio de las predicciones individuales de cada clasificador, minimizando de esta manera el porcentaje de error. Sin embargo, el modelo basado en el ensamble Gradient Boosted Tree el cual también utiliza árboles de decisión como clasificadores débiles sobre el conjunto de datos completo, por su naturaleza de retroalimentación secuencial a partir de los errores identificados en los clasificadores previos, presentó mejor resultado en cuanto a precisión. Por lo anterior, es importante mencionar que una solución basada en un ensamble de clasificadores en el escenario utilizado, ofrece una mejor solución y a partir de los resultados obtenidos se puede tomar la decisión de utilizar determinado ensamble considerando un balance entre rapidez y precisión a partir de las necesidades específicas del entorno.

Los resultados obtenidos de la evaluación del desempeño de cada modelo se muestran en la gráfica de la figura 2.

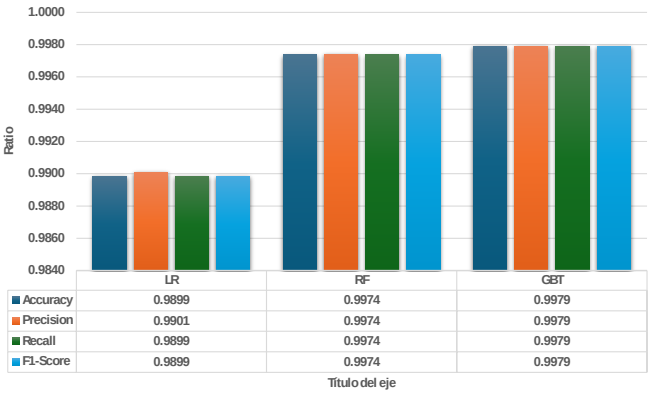


Figura 2. Desempeño de los modelos basados en ML utilizados para la detección de ataques DDoS.

Los resultados con el tiempo de ejecución de cada modelo se muestran en la tabla V.

Tabla V
TIEMPOS DE EJECUCIÓN

Modelo ML	***Tiempo de ejecución (seg)***
LR	57.8
RF	58.3
GBT	85.5

En las figuras 3, 5 y 4 se muestran las matrices de confusión del rendimiento de los modelos basados en ML que fueron utilizados, a través de las cuales fue posible realizar la comparativa de la eficacia de cada modelo en la detección de ataques TCP-SYN.

VI. DISCUSIÓN

El desarrollo de la implementación de un clúster para el entrenamiento de modelos basados en ML para la detección de ataques DDoS puede ser considerado como una alternativa eficiente y escalable para líneas de investigación que requieren capacidades de cómputo de alto desempeño y escalabilidad, pero es importante mencionar que el trabajo realizado presenta diversas áreas de oportunidad. Una limitación identificada es

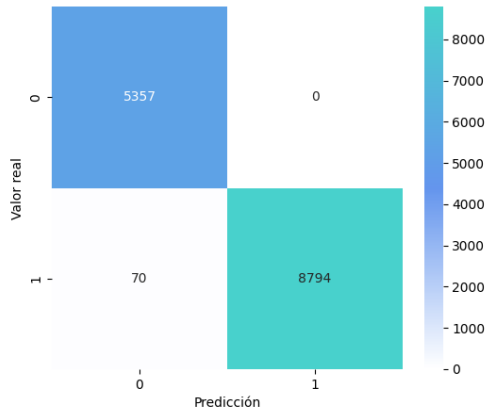


Figura 3. Regresión Logística (SYN).

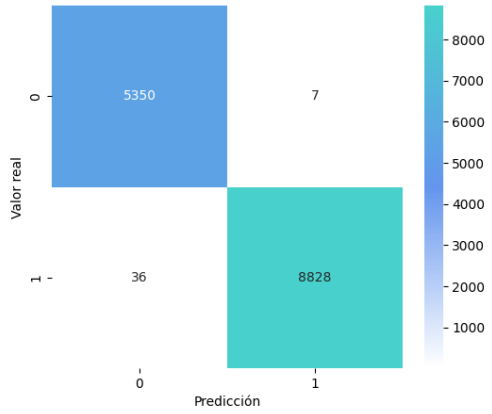


Figura 4. Gradient Boosted Tree (SYN).

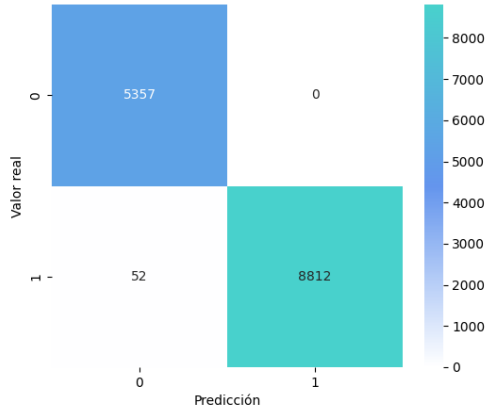


Figura 5. Random Forest (SYN).

que solo se utilizó el dataset CICDDoS-2019, que si bien incluye varios tipos de ataque DDoS este no incluye datos sobre ataques DDoS más recientes, como los de tipo low rate o de múltiples vectores. Otras limitaciones en el conjunto de datos es que no está balanceado lo que puede generar una tasa alta de falsos positivos y que su falta de diversidad de ataques DDoS puede provocar sobreajuste en los modelos. Adicionalmente, es importante mencionar que se utilizaron solamente tres modelos basados en ML lo cual representa solo una muestra representativa de los modelos existentes basados en ML. En el contexto de ciberseguridad se ha identificado que un área de oportunidad para los mecanismos de seguridad que utilizan modelos basados en ML es lograr que sus soluciones sean escalables para el almacenamiento de los conjuntos de datos que utilizan y para mejorar el tiempo procesamiento y validación. Por lo anterior, se plantea que el trabajo realizado podría ser utilizado como plataforma base para otras líneas de investigación en ciberseguridad como la detección de malware, para la generación y almacenamiento de conjuntos de datos a partir de redes generativas antagónicas (GAN's) la cual es un área muy relevante debido a la falta de conjuntos de datos grandes y diversos disponibles para el entrenamiento de los modelos y para la detección de amenazas en ecosistemas específicos como Internet de las Cosas, Redes Definidas por Software o Cómputo en la Nube. El presente trabajo puede evolucionar hacia las líneas de investigación mencionadas previamente a través del uso de conjuntos de datos especializados en esas áreas y mediante la integración al escenario del clúster de una arquitectura basada en microservicios que permita realizar el despliegue rápido y escalable de múltiples modelos basados en ML para la detección de diversos ataques cibernéticos de forma simultánea.

VII. Conclusión

En este trabajo se presenta la implementación de un clúster para la fase de entrenamiento de modelos basados en ML para la detección de ataques DDoS, con capacidades de almacenamiento y procesamiento escalables mediante cómputo distribuido. Durante el desarrollo del trabajo fue posible evaluar el desempeño de los modelos LR, RF y GBT para la detección de ataques DDoS de tipo Reflectivo (DNS y NTP) y por Inundación (TCP SYN Flood y UDP Flood). El entrenamiento de los modelos fue realizado a partir de las características más relevantes de cada uno de los ataques DDoS seleccionados. Los resultados de la experimentación demostraron que la fase de entrenamiento de los modelos utilizados se realizó en el orden de 57 a 86 segundos y que el modelo que obtuvo mejores resultados con respecto al balance entre su efectividad y tiempo de ejecución fue Random Forest, mientras que el modelo Gradient Boosted Tree obtuvo los mejores resultados en cuanto a precisión. El escenario para cómputo distribuido tipo clúster implementado demostró que puede ser una alternativa para proveer de forma transparente paralelización de tareas, tolerancia a fallos, redundancia y balanceo de carga para mecanismos de detección de ataques DDoS que utilizan modelos basados en ML, por lo cual puede

ser una plataforma que sirva de base para el despliegue de estos mecanismos un ambiente de producción.

VIII. Trabajo futuro

La implementación del clúster si bien provee requerimientos no funcionales de escalabilidad y tolerancia a fallos también presenta áreas de mejora. Un área de oportunidad identificada es la necesidad de complementar la arquitectura del clúster con un módulo para captura de tráfico y un componente que permita utilizar los modelos basados en ML que ya fueron entrenados para realizar detección de ataques DDoS en tiempo real, así como incluir un componente para monitoreo y actualización del entrenamiento de los modelos para detectar ataques DDoS más recientes. Una línea de investigación que también se considera como trabajo futuro es utilizar el clúster como componente de procesamiento para realizar experimentación con modelos de Deep Learning para la detección de ataques DDoS de día cero. Para complementar la evaluación del clúster en la detección de ataques DDoS en un escenario real, se considera utilizar métricas que permitan evaluar su desempeño considerando las variables del entorno como: uso de CPU, uso de memoria RAM, ancho de banda, así como métricas específicas de ciberseguridad como: Mean Time To Detect (MTTD) y Mean Time To Respond (MTTR).

IX. Agradecimientos

El presente trabajo fue realizado en el marco del curso de Sistemas Distribuidos del posgrado en Ciencias y Tecnologías de Seguridad, por lo cual se agradece especialmente a la profesora del curso por su asesoría y retroalimentación. Se extiende el agradecimiento a la SECIHTI por el apoyo brindado a través de la beca para el estudio del posgrado, al INAOE por las facilidades otorgadas para realizar estudios de posgrado en un entorno de excelencia académica, a los profesores del núcleo académico por sus enseñanzas, así como a los autores de las publicaciones y de los recursos utilizados.

Referencias

[1] Abdallah, Amira Mahamat, Aysha Saif Rashed Obaid Alkaabi, Ghaya Bark Nasser Douman Alameri, Saida Hafsa Rafique, Nura Shifa Musa y Thangavel Murugan: *Cloud Network Anomaly Detection Using Machine and Deep Learning Techniques— Recent Research Advancements*. IEEE Access, 12:56749–56773, 2024.

[2] Coulouris, George, Jean Dollimore, Tim Kindberg y Gordon Blair: *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edición, 2011, ISBN 0132143011.

[3] Dai, Fei, Md Akbar Hossain y Yi Wang: *State of the Art in Parallel and Distributed Systems: Emerging Trends and Challenges*. Electronics, 14(4), 2025, ISSN 2079-9292. https://www.mdpi.com/2079-9292/14/4/677.

[4] Groenewald, Elma Sibonghanoy, Sarath Babu Dodda, Coenrad Adolph Groenewald, Amol Dhumane, Aditi Sharma y Ketan Kotecha: *Parallelized Distributed Computing for Large-Scale Support Vector Machine Training*. En Goar, Vishal, Aditi Sharma, Jungpil Shin y M. Firoz Mridha (editores): *Deep Learning and Visual Artificial Intelligence*, páginas 445–460, Singapore, 2024. Springer Nature Singapore, ISBN 978-981-97-4533-3.

[5] Hameed, Sufian y Usman Ali: *Efficacy of Live DDoS Detection with Hadoop*. En *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, páginas 488–494, 2016.

[6] Ivanova, Petya y Todor Tagarev: *Challenges and Opportunities for Network Intrusion Detection in a Big Data Environment*. En Tagarev, Todor y Nikolai Stoianov (editores): *Digital Transformation, Cyber Security and Resilience*, páginas 93–106, Cham, 2024. Springer Nature Switzerland, ISBN 978-3-031-44440-1.

[7] Jawad, Wasnaa y Abbas Al-Bakry: *Performance evaluation of Hadoop and spark distributed frameworks by using them to build an intelligent intrusion detection system*. En *American Institute of Physics Conference Series*, volumen 3211 de *American Institute of Physics Conference Series*, página 030005, Mayo 2025.

[8] Ouhssini, Mohamed, Karim Afdel, Mohamed Akouhar, Elhafed Agherrabi y Abdallah Abarda: *Advancements in detecting, preventing, and mitigating DDoS attacks in cloud environments: A comprehensive systematic review of state-of-the-art approaches*. Egyptian Informatics Journal, 27:100517, 2024, ISSN 1110-8665. https://www.sciencedirect.com/science/article/pii/S111086652400080X.

[9] Raparthi, Mohan, Monika Soni, Vipin Tiwari, Amol Dhumane y Rahul Sharma: *Scalable Implementation of Random Forests for Big Data Classification on Cloud Infrastructure*. En Goar, Vishal, Aditi Sharma, Jungpil Shin y M. Firoz Mridha (editores): *Deep Learning and Visual Artificial Intelligence*, páginas 493–512, Singapore, 2024. Springer Nature Singapore, ISBN 978-981-97-4533-3.

[10] Sharafaldin, Iman, Arash Habibi Lashkari, Saqib Hakak y Ali A. Ghorbani: *Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy*. En *2019 International Carnahan Conference on Security Technology (ICCST)*, páginas 1–8, 2019.

[11] Steen, Maarten y Andrew S. Tanenbaum: *A brief introduction to distributed systems*. Computing, 98(10):967–1009, Octubre 2016, ISSN 0010-485X. https://doi.org/10.1007/s00607-016-0508-7.

[12] Vashishtha, Lalit Kumar y Kakali Chatterjee: *Strengthening cybersecurity: TestCloudIDS dataset and SparkShield algorithm for robust threat detection*. Computers & Security, 151:104308, 2025, ISSN 0167-4048. https://www.sciencedirect.com/science/article/pii/S016740482400614X.

Principales enfoques y métodos para la detección de iSQL

Stanislav Glebskiy
Instituto Nacional de Astrofísica Óptica y Electrónica
Puebla, México
stanislav.glebskiy@inaoep.mx

Dra. Lil María Xibai Rodríguez Henríquez,
Instituto Nacional de Astrofísica Óptica y Electrónica
Puebla, México
lmrodriguez@inaoep.mx

***Abstract*—Las inyecciones de SQL son un tema de gran relevancia en la protección de datos digitales almacenados en bases de datos relacionales. En este artículo se habla de manera general sobre las tendencias en el desarrollo de métodos para detectar y prevenir este tipo de ataques, enfocándose tanto en el sector industrial como en el ámbito de investigación académica. Se presentan de forma simplificada los conceptos necesarios para entender el tema. Este artículo pretende servir de introducción a cualquier persona que quiera adentrarse en el mundo de las iSQL (inyecciones SQL) y su mitigación, presentando un panorama general del área y discutiendo los enfoques más recientes de investigación.**

***Palabras clave*—Inyecciones SQL, bases de datos, seguridad.**

I. Introducción

Gran parte de la información que se maneja en la sociedad se almacena hoy en día de manera digital. Un método de almacenamiento de información digital son las Bases de Datos Relacionales. Estas son implementadas a través de Sistemas de Manejo de Bases de Datos Relacionales o RDBMS (por sus siglas en inglés [13]). El mantenimiento, uso y gestión adecuados de las bases de datos son relevantes para las empresas e instituciones, ya que les permiten tomar decisiones informadas y mantener una operación eficiente al salvaguardar los datos de la empresa de manera segura, disponible e íntegra.

Cualquier persona, institución o empresa que pretenda mantener una base de datos debe garantizar la seguridad de la información que contiene. Existen retos de seguridad específicos que conciernen a las Bases de Datos Relacionales. Uno de los retos más destacados es la detección y prevención de iSQL. Los ataques de inyección de SQL son una de las vulnerabilidades más comunes y peligrosas que afligen a los sistemas que contienen bases de datos. Según la OWASP, estos ataques figuraban como el ataque más común en el año 2017 y el tercero más común en el año 2021 [1]. Utilizando iSQL, un usuario malintencionado puede acceder a información confidencial o incluso llegar a alterar la información de la base de datos.

En este artículo se pretende dar una introducción simplificada al tema de las inyecciones SQL y las diferentes maneras de mitigarlas. Se introducen los conceptos esenciales necesarios para entender las iSQL como el entorno en el que ocurren, cómo se definen y cómo se clasifican. Posteriormente, se procede a exponer las soluciones que se han propuesto para mitigar estas vulnerabilidades, distinguiendo dos ramas principales: soluciones que se ven en entornos industriales y soluciones que están siendo desarrolladas en el ámbito académico. Para ilustrar los ejemplos presentados más adelante se toma como referencia una base de datos denominada ”empresa” presentada en la Figura 1 que está compuesta por dos tablas: ”almacén” y ”empleados”.

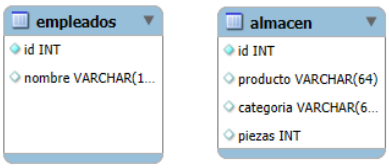


Fig. 1. Base de datos ”empresa”

En la sección II se explica qué es una inyección SQL, en la sección III se mencionan las principales categorías de iSQL que existen y se presentan algunos ejemplos, en la sección IV se habla del entorno donde se producen las iSQL y donde se pueden mitigar, en la sección V se habla de los principales enfoques de mitigación de iSQL utilizados en sistemas de producción, mientras que en la sección VI se mencionan los enfoques que se pueden encontrar en el mundo académico, en la sección VII se presenta la discusión donde se comparan los enfoques industriales y académicos y se detallan las tendencias actuales del campo, por último, en la sección VIII se muestran las conclusiones del artículo.

II. ¿Qué es una iSQL?

SQL (del inglés Structured Query Language) es el lenguaje que se utiliza para interactuar con las bases de datos. Tanto aplicaciones como personas pueden utilizar SQL. SQL son los comandos que un RDBMS puede entender y ejecutar. Por lo general, las aplicaciones web utilizan datos

proporcionados por los usuarios para formular comandos SQL.

Una iSQL se puede definir como la alteración de un comando SQL para lograr un comportamiento de la base de datos diferente al normal mediante el uso de entradas de usuario que se concatenan al comando SQL [13].

Se propone un ejemplo donde se tiene una aplicación web que se dedica a la venta de artículos por internet. Para llevar la gestión de todos sus productos, ventas, usuarios, etc., la aplicación web utiliza la base de datos relacional ”empresa” de la Figura 1. Cuando un cliente hace una consulta, selecciona el tipo de artículos que le interesa de una lista disponible en la página web. La página web recibe esta información en forma de una cadena de caracteres y genera un comando SQL para que el RDBMS le entregue todos los artículos relevantes que existen en la base de datos. En la Figura 2 se muestra este sistema en funcionamiento con un cliente seleccionando la categoría de ”ropa”. Así se vería un comando SQL benigno generado en la situación descrita en el ejemplo anterior:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa';
```

Pero el hecho de que la aplicación web maneje las categorías como cadenas de caracteres y las concatene directamente en el comando SQL abre la puerta a las iSQL. Si los usuarios proporcionan palabras o caracteres que tienen un significado en el lenguaje SQL, la función del comando resultante puede ser alterada. Un comando SQL alterado por una iSQL se vería como:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or 1=1 ;-- ';
```

En este caso el usuario logró modificar la información que se envía al servidor web para que la categoría fuera “ropa’ or 1=1 ;– ”, en vez de ”ropa”. Esta entrada alteraría la manera en la que el RDBMS procesa el comando SQL. En el primer caso el RDBMS devolvería todos los productos de la categoría ”ropa”, pero en el segundo caso expondría todos los productos en la tabla ”almacen”. Esto podría no ser deseado si la tabla contiene categorías de productos que no deberían estar expuestas a los clientes.

Aunque las iSQLs se conocen desde mediados de los años 90s, hasta el día de hoy, es un tema que no se ha resuelto en su totalidad. Se han propuesto muchas soluciones, pero es difícil encontrar una que abarque todos los tipos de iSQL y ambientes de implementación, ya que como se verá en la Sección III, estos ataques son muy variados.

La razón por la que es difícil prevenir las iSQL es porque, de acuerdo con Su Z. et. al. [2] existe una brecha semántica entre la aplicación web que genera el comando y la base de datos que interpreta el comando. Es decir, la aplicación web no entiende propiamente el comando SQL, para la aplicación web, el comando SQL solo es una cadena de caracteres.

III. Tipos de iSQLs

Existen varios tipos de iSQL. A continuación, mencionamos los tipos de iSQL más conocidos categorizados por cómo se altera el comando SQL propuesto por Devi R. et. al. en [19] .

A. Tautologías

Se basan en la cláusula WHERE del lenguaje SQL. Son aquellas donde se inyecta una condición que siempre es cierta en la cláusula WHERE de un comando SELECT. De esta forma, se logra invalidar los filtros de una búsqueda SQL y obtener información restringida. En la sección II, se mostró un ejemplo de un iSQL de este tipo. Esta inyección le permitiría a un atacante obtener todo el contenido de la tabla ”almacén” de la Figura 1.

B. Unión

Son aquellas donde inyectando la cláusula UNION en un comando SELECT el atacante es capaz de obtener información de tablas o columnas adicionales. A continuación se muestra un ejemplo de esta iSQL:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' UNION
  SELECT nombre
  FROM empleados;-- ';
```

Esta inyección le permitiría a un atacante acceder al contenido de la tabla ”empleados” de la Figura 1.

C. Piggyback query

Son aquellas donde, mediante la inyección de caracteres de fin de comando, como “;”, el atacante es capaz de agregar un segundo comando SQL para que este sea ejecutado junto al comando original. A continuación se muestra un ejemplo de esta iSQL:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa';
  DROP DATABASE empresa;-- ';
```

D. Basadas en errores

Son aquellas donde se extrae información del sistema de base de datos al provocar la ejecución de un comando SQL mal formado. La base de datos retorna un mensaje de error y el atacante puede obtener información del sistema a partir de este.

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' WHERE;-- ';
```

De igual manera, de acuerdo a Rai A. et. al. en [18], las iSQL se pueden categorizar por la manera en la que se ejecutan:

E. Clásicas

Son aquellas donde el atacante obtiene la respuesta de un intento de inyección por el mismo canal por donde se realiza el ataque. Este es el tipo de iSQL más común. Un ejemplo sería cuando un atacante manda inyecciones SQL a través de una página web y observa la respuesta del sistema en la misma página web. Todos los ejemplos de iSQL presentados anteriormente se pueden ejecutar de esta forma.

F. De segundo orden

Se dan cuando el atacante logra almacenar un comando SQL manipulado directamente en la base de datos. Este comando malicioso es ejecutado en el futuro. En este caso, la entrada maliciosa no proviene directamente del usuario, sino de la base de datos misma. Por ejemplo, el atacante logra agregar un nuevo producto a la tabla "almacén" de la base de datos de la Figura 1 con el siguiente comando SQL:

```
INSERT INTO almacen (id,
  producto,
  categoria,
  piezas)
VALUES (6,
  'producto6',
  'ropa''; DROP DATABASE empresa ',
  1);
```

Posteriormente, si algún programa lee la columna "categoría" de este nuevo producto y la concatena a otro comando SQL, puede provocar una iSQL de tipo "Piggyback query" y eliminar toda la base de datos.

G. Ciegas

Son aquellas donde el atacante no es capaz de visualizar una respuesta directa del RDBMS, pero puede manipular algún componente del sistema. El ejemplo más común es cuando un atacante logra ejecutar una cláusula SLEEP() en el RDBMS y, midiendo el tiempo de respuesta del sistema, puede saber que su ataque está siendo ejecutado en el RDBMS. A continuación se presenta un ejemplo:

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or
  SLEEP(10);-- ';
```

H. Out-of-bound

Son aquellas donde el canal por donde se envía el ataque es diferente al canal por donde se recibe la respuesta. Por ejemplo, el atacante inyecta un SQL malicioso en una página web que hace que el RDBMS guarde información sensible en un archivo del SO (sistema operativo). Posteriormente, el atacante accede por otro medio al SO y descarga el archivo, obteniendo así la información de la base de datos.

```
SELECT producto
FROM almacen
WHERE categoria = 'ropa' or 1=1 INTO
  OUTFILE '/tmp/myfile';-- ';
```

IV. Entorno de iSQL

Cuando se hace referencia a las iSQL generalmente se considera el contexto de aplicaciones web. Si bien, en muchas ocasiones estas aplicaciones llegan a tener arquitecturas de hardware y software complejas, se pueden simplificar a una arquitectura de tres capas [13] que nos permite entender el flujo de datos y cómo se generan las peticiones SQL. En la Figura 2 podemos distinguir los elementos que conforman este modelo:

1) Los clientes
2) El servidor de la aplicación web
3) El servidor de la base de datos

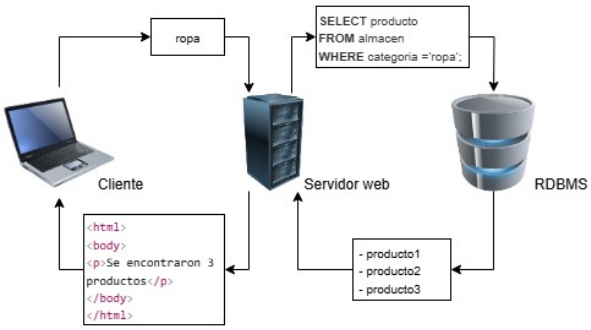


Fig. 2. Arquitectura web de tres capas.

En esta arquitectura, los comandos SQL son generados por la aplicación web (que reside en el servidor web) a partir de la información enviada por el cliente. El resultado del comando SQL es utilizado para devolver la información necesaria al cliente, por lo general como contenido de una página web. En la Figura 2 se observa el flujo de información, empezando por el cliente haciendo una petición que llega al servidor web. Posteriormente, la aplicación web hace una consulta a la base de datos utilizando lenguaje SQL. Por último, la información devuelta por la base de datos es utilizada para crear el contenido de la página web que se le devuelve al cliente.

Un sistema de detección de intrusión IDS (por sus siglas en inglés) de iSQL se puede implementar en cada uno de los elementos de una arquitectura de tres capas y también en los canales de comunicación entre estos elementos. Por lo que, de acuerdo a Kar D. et. al. [3], podemos definir 5 niveles en los que se puede implementar un IDS de iSQL:

1) En las aplicaciones de los clientes.
2) Entre los clientes y la aplicación web (firewall de la aplicación web).
3) En la aplicación web.
4) Entre la aplicación web y el RDBMS (firewall del RDBMS).
5) En el RDBMS.

Por lo general, una misma base de datos suele ser usada por varias aplicaciones web. Y una aplicación web suele tener varios clientes. Por lo que, si un IDS de iSQL se implementa en un nivel más bajo, se tiene que implementar en más instancias. Considerando el caso de una única base de datos que sirve a tres aplicaciones web distintas. Basta con que una sola aplicación web sea vulnerable a iSQL para que toda la base de datos quede comprometida.

Cuando se habla de soluciones a las iSQL se puede distinguir entre el sector industrial, donde se implementan soluciones maduras en ambientes de producción, o el sector académico, donde se proponen y estudian soluciones más innovadoras, pero menos maduras.

V. Soluciones industriales

Los IDS de iSQL que se implementan en ambientes de producción tienen que ser sistemas ampliamente probados y que no tengan efectos negativos como ralentizar el procesamiento de las bases de datos o aplicaciones web. Los principales enfoques para solucionar las iSQL en ambientes industriales son:

A. Expresiones regulares

Estos son mecanismos que detectan patrones comunes de iSQL mediante el uso de expresiones regulares. Estos mecanismos se encuentran comúnmente implementados en los firewalls a distintos niveles en la arquitectura web. Por lo general, los firewalls de red y de aplicación web (por sus siglas en inglés WAF) no han tenido un buen desempeño en la detección de ataques de iSQL [16]. Los firewalls de bases de datos suelen tener mejor desempeño en la detección de ataques de iSQL. Aun así, los sistemas IDS de iSQL basados en expresiones regulares no son buenos para detectar iSQL complejas y pueden ser engañados con técnicas de codificación del ataque iSQL [15].

B. Listas negras y blancas

Las listas negras son conjuntos de comandos SQL o expresiones que el sistema bloquea. Las listas blancas, por el contrario, son listas de comandos SQL que el sistema deja pasar, es decir, bloquea todos los comandos SQL a excepción de los que están en la lista blanca. Ejemplos de sistemas que utilizan esta técnica se pueden ver en [5] y [6].

C. Verificación de código de aplicaciones web

Una aplicación web que está correctamente diseñada y cuenta con una buena implementación, no deja pasar iSQL. Algunas de las recomendaciones que se deben seguir al diseñar aplicaciones web que trabajen con bases de datos [15] son:

- Validación de datos proporcionados por los usuarios.
- Uso de comandos SQL parametrizados.
- Uso de procedimientos de RDBMS almacenados.
- Codificación de los datos.

Sidik R. F. et. al. [17] demostró la eficacia del uso de comandos SQL parametrizados en el lenguaje Golang al mitigar todas las vulnerabilidades de iSQL en su ambiente de experimentación. Estos métodos se implementan mediante la capacitación de los desarrolladores web. De igual manera, existen analizadores de código que pueden detectar vulnerabilidades de iSQLs y sugerir algunos de los métodos presentados anteriormente para mejorar el código fuente.

D. Configuración segura del entorno

Por último, el seguimiento de normas y estándares de seguridad es la principal defensa contra las iSQL. Estos incluyen:

- Segmentación de los datos.
- Cifrado de datos sensibles.
- Manejo adecuando de usuarios.
- Principio del mínimo privilegió.

Estos se deben aplicar tanto durante el desarrollo de las aplicaciones y bases de datos como durante su ejecución en ambientes de producción. La configuración correcta de un sistema elimina gran parte de las posibles vulnerabilidades. Como ejemplo, Hamidi A. [7] explora los diferentes mecanismos que interactúan en el RDBMS de MySQL para proporcionar seguridad a la base de datos y habla sobre la correcta configuración de estos.

VI. Soluciones académicas

En el ámbito académico se encuentra una mayor variedad de enfoques y mecanismos para la mitigación de las iSQL. Cabe mencionar que todas las soluciones utilizadas en el sector industrial se siguen estudiando y mejorando en el mundo académico, pero también se pueden observar enfoques nuevos.

A. Comparación de árbol de parseo

Un enfoque interesante es la comparación de los árboles de parseo de un comando SQL antes y después del input de los usuarios propuesto por Buehrer G. et. al. en [8]. Cuando un comando SQL es procesado por un RDBMS pasa por varias etapas. Una de ellas es la transformación del comando SQL en un árbol donde cada elemento del comando es representado por un nodo. Diferentes comandos SQL van a resultar en diferentes árboles. Una misma estructura de un árbol puede

contener distintos datos. Es decir, que la estructura del árbol representa el esqueleto del comando SQL. Si un comando SQL es alterado, su funcionalidad cambia y va a producir un árbol diferente. Otro ejemplo de este enfoque es presentado por Katole R. et. al. en [9].

B. Métodos basados en tokens

Este enfoque, observado por Shahbaaz M en [24], se centra en expresar un comando SQL mediante una serie de tokens predefinidos. De esta forma, se logra transformar el comando original, eliminando información irrelevante y preservando las características esenciales del comando SQL. Analizar los tokens simplifica la identificación de patrones de iSQL. Los trabajos de Kar D. et. al. en [4] y [3] son buenas ejemplificaciones de este enfoque.

C. Métodos basados en comportamiento

Por otro lado, existe el enfoque basado en detectar comportamiento anómalo para prevenir ataques de iSQL. Estos métodos se basan en establecer un modelo de lo que es "normal" para un sistema y, cuando detectan que algo sale de ese patrón, lo clasifican como peligroso. Estos métodos son similares en su esencia a los sistemas UEBA (del sus siglas en inglés User and Entity Behavior Analytics), donde se modela el comportamiento de usuario y se determina si una acción es "normal" o "anormal".

Esta es una gama bastante amplia de soluciones, ya que se pueden implementar de distintas maneras y en distintos niveles de los sistemas web. Medeiros I. et. al. [10] presenta un sistema implementado dentro del mismo RDBMS, mientras que Fonseca J. et. al. [11] presenta un mecanismo implementado como Firewall del RDBMS. Algo característico de este enfoque es que necesita de una fase de aprendizaje. Es decir, existe un intervalo de tiempo donde el sistema debe aprender lo que se clasifica como peligroso y lo que no lo es. El aprendizaje se puede hacer con distintos métodos, pero sobresalen dos categorías más comunes:

- El aprendizaje se hace durante el funcionamiento normal del sistema. En este caso el sistema aprende lo que es el funcionamiento "normal" y si posteriormente detecta algo que no haya visto, lo marca como "anormal".
- Se cuenta con un conjunto de entradas peligrosas y no peligrosas. En este caso el administrador es encargado alimentar el sistema con las entradas ya marcadas.

Por lo general, las soluciones basadas en comportamiento son dependientes de su entorno de implementación. Es decir, su funcionamiento depende del ambiente en el que se estén ejecutando. Esto tiene sentido, ya que los ataques de iSQL se ajustan para cada sistema específico. Lo que puede ser una iSQL en un entorno puede no serlo en otro.

D. Machine Learning e inteligencia artificial

El auge de las tecnologías de Machine Learning e Inteligencia Artificial ha impulsado mucho la rama de IDS de iSQL [12]. Mientras que la idea general de este enfoque es similar a las soluciones basadas en comportamiento: aprender a diferenciar entre entradas "normales" y "anormales" a partir de un conjunto de datos, ahora se cuenta con herramientas matemáticas mucho más potentes para lograr este objetivo. En los últimos años se ha observado un gran número de trabajos académicos donde se han probado distintos modelos y algoritmos de Machine Learning para la detección de iSQL. Lu Z. [21] propone un clasificador Naive Bayes para detectar iSQL, Shahbaz M. et. al. [22] proponen un sistema basado en redes neuronales convolucionales que logra una precisión de 97%. Por otro lado, Shakya et. al. [23] en su trabajo implementaron sistemas de detección de iSQL usando algoritmos de SVM (del inglés Support Vector Machine), KNN (de inglés K-Nearest Neighbor) y RF (del inglés Random Forest) logrando precisiones de 98.28%, 99.55% y 87.72% respectivamente.

Maha et. al. [14] presenta la comparación de 36 propuestas de sistemas de detección de iSQL basados en Machine Learning publicados entre los años 2012 y 2021. Frecuentemente, los algoritmos de Machine Learning se usan para mejorar soluciones ya existentes. Babaey V. et. al. [20] propone un sistema basado en inteligencia artificial para generar reglas de WAF más eficientes.

VII. Discusión

Se observa una diferencia significativa entre las soluciones industriales y las propuestas académicas en el ámbito de detección de iSQL. Los sistemas de producción suelen utilizar métodos más básicos, pero bien probados y pulidos a través de varias décadas. Estos sistemas suelen cubrir los intentos de ataques de iSQL más comunes. En la Sección V se presentaron los enfoques de uso de expresiones regulares, listas negras y blancas, verificación de código de aplicaciones web y configuración segura del entorno. La verificación de código de la aplicación web y la configuración segura del entorno son la primera línea de defensa contra los ataques de iSQL y los que más intentos de iSQL previenen. Esto se debe a que, a diferencia de los otros dos métodos, previenen los ataques desde antes de que ocurran, mientras que los demás métodos mitigan el ataque cuando ya está en ejecución.

Los enfoques académicos modernos, si bien, muestran resultados prometedores, suelen ser soluciones bastante complejas y costosas. Surge la pregunta, ¿cuánto está dispuesto a invertir una empresa en un sistema IDS de iSQL? Si un sistema IDS de iSQL es demasiado caro o difícil de operar, muchas empresas no tendrán la posibilidad de implementarlo.

En la Sección VI se expusieron los enfoques de comparación de árbol de parseo, métodos basados en

tokens, métodos basados en comportamiento y los métodos basados en Machine Learning e inteligencia artificial, este último siendo una rama que se está desarrollando muy activamente en el ámbito académico. Este enfoque es bastante prometedor ya que, de acuerdo a Maha et. al. [14], las propuestas de IDS de iSQL que utilizan modelos de Machine Learning por lo general alcanzan porcentajes de detección arriba del 90%. Pero estas soluciones suelen ser computacionalmente costosas y requieren una etapa de aprendizaje. Aparte, esta etapa de aprendizaje suele atar el sistema de IDS de iSQL a un entorno específico, es decir, si se requiere modificar el entorno de ejecución del sistema IDS, se debe repetir el proceso de aprendizaje.

La tendencia más actual de la investigación académica está enfocada en identificar cómo se pueden utilizar los avances que se han hecho en el campo de la inteligencia artificial y Machine Learning en la detección y prevención de ataques iSQL. Se están probando distintos algoritmos de Machine Learning y redes neuronales para determinar cuáles son los mejores para identificar iSQL.

La solución más efectiva continúa siendo un diseño y programación seguros de aplicaciones web. Esto debido a que siempre es más fácil evitar un ataque que identificarlo y detenerlo en el momento de su ejecución. El problema de este enfoque es que da paso al error humano durante el proceso de desarrollo de aplicaciones web. Otra desventaja es que no siempre se tiene acceso al código fuente vulnerable. Aplicar estos métodos no es posible en software de terceros. Por lo que los sistemas IDS de iSQL siguen siendo necesarios.

VIII. Conclusión

Aún no se ha propuesto una solución definitiva a las iSQL. Existen varios enfoques para tratar de solucionar este problema. Por lo general, los métodos más probados y maduros se utilizan en la industria en sistemas de producción, mientras que el mundo académico está buscando enfoques más innovadores para atacar las iSQL. Las tendencias hacia la IA han marcado su huella en el diseño de nuevas soluciones a las iSQL. La detección y prevención de iSQL es un campo de investigación maduro, pero aún no agotado, ya que se están proponiendo nuevos enfoques para combatir la vulnerabilidad de las iSQL.

References

[1] OWASP, Owasp top ten, https://owasp.org/www-project-top-ten/, Visto: 2024-05-10.

[2] Zhendong Su, Gary Wassermann, "The essence of command injection attacks in web applications", 2006, http://dx.doi.org/10.1145/1111320.1111070.

[3] Debabrata Kar, Suvasini Panigrahi, Srikanth Sundararajan, "SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM", Computers & Security, Volumen 60, 2016, https://doi.org/10.1016/j.cose.2016.04.005.

[4] Debabrata Kar, Suvasini Panigrahi, Srikanth Sundararajan, "SQLiDDS: SQL Injection Detection using Query Transformation and Document Similarity", 11th International Conference on Distributed Computing and Internet Technology, 2015, http://dx.doi.org/10.1007/978-3-319-14977-6_41

[5] Oracle, Oracle SQL Firewall User's Guide, 2025, https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlfw/overview-oracle-sql-firewall.html

[6] Oracle, MySQL Enterprise Firewall, 2024, https://www.mysql.com/products/enterprise/firewall.html

[7] Abdullah Hamidi, Abdul Razzaq Hamraz y Khadija Rahmani, "Database Security Mechanisms in MySQL", Afghanistan Research Journal - Natural Science, Volumen 4, 2022

[8] Gregory T. Buehrer, Bruce W. Weide y Paolo A. G. Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks", Proceedings of the 5th International Workshop on Software Engineering and Middleware, 2005

[9] Rajashree A. Katole, Swati S. Sherekar y Vilas M. Thakare, "Detection of SQL Injection Attacks by Removing the Parameter Values of SQL Query", Proceedings of the Second International Conference on Inventive Systems and Control, 2018.

[10] Ibéria Medeiros, Beatriz Miguel, Nuno Neves y Miguel Correia, "Hacking the DBMS to Prevent Injection Attacks", 2016

[11] José Fonseca, Marco Vieira y Henrique Madeira, "Detecting Malicious SQL", TrustBus, Volumne 4657, 2007

[12] Mohammed Nasereddin, Malik Qasaimeha, Ashaar Alkhamaiseh y Raad Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks", Information Security Journal: A Global Perspective, Volumen 3548, 2021

[13] Ramez Elmasri y Shamkant B. Navathe, "Fundamentals of Database Systems", Pearson, 2016

[14] Maha Alghawazi, Daniyal Alghazzawi y Suaad Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review", Journal of Cybersecurity and Privacy

[15] Justin Clarke, "SQL Injection Attacks and Defense", Elsevier, 2009

[16] Oracle, "Mitigating Risks of SQL Injection", 2023

[17] Rizaldi Fatah Sidik, Syifa Nurgaida Yutia y Rana Zaini Fathiyana, "The Effectiveness of Parameterized Queries in Preventing SQL Injection Attacks at Go", Proceedings of the International Conference on Enterprise and Industrial Systems, 2023

[18] Aditya Rai, Mazharul Islam Miraz, Deshbandhu Das, Harpreet Kaur y Swati, "SQL Injection: Classification and Prevention", International Conference on Intelligent Engineering and Management, 2021

[19] Rubidha Devi, Raghuraman Koteeswaran y Ramasamy Venkatesan, "A study on SQL injection techniques", International Journal of Pharmacy and Technology, 2016

[20] Vahid Babaey y Arun Ravindran, "GenSQLi: A Generative Artificial Intelligence Framework for Automatically Securing Web Application Firewalls Against Structured Query Language Injection Attacks", Future Internet, 2025

[21] Zhexi Lu, "Sql injection detection using Naïve Bayes classifier: a probabilistic approach for web application security", ITM Web of Conferences, 2025, https://doi.org/10.1051/itmconf/20257004016

[22] Muhammad Shahbaz, Gohar Mumtaz, Saleem Zubair y Mudassar Rehman, "Evaluating CNN Effectiveness in SQL Injection Attack Detection", Journal of Computing & Biomedical Informatics, 2024, https://doi.org/10.56979/702/2024

[23] Shakya, Dharmaratne y Manjula Sandirigama, "Detection of SQL Injection Attacks Using Machine Learning Techniques", 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE), 2024, https://doi.org/10.1109/ICECCE63537.2024.10823462

[24] Shahbaaz Mohammed Hayat Chaki, "A Comparative Analysis of SQL Injection Prevention Methods: Evaluating SQLPMDS vs SIUQAPTT Effectiveness", TechRxiv, 2025

Índice de Autores