Cifrado de Datos Dinámico con Inteligencia Artificial, Utilizando el Nuevo Formato Pseudo-Hexadecimal

[1]Edgar Rangel Lugo ,[1]Kevin Uriel Rangel Ríos , [1]Leonel González Vidales

[1]Tecnológico Nacional de México. I. T. Ciudad Altamirano.
Depto. Sistemas y Computación
erangel_lugo@hotmail.com

Recibido*: 22 de noviembre de 2024*
Aceptado*:17 de diciembre de 2024*

RESUMEN

El robo digital de datos, ocasiona pérdidas en las finanzas de las organizaciones, cuando se utilizan estrategias de ciberseguridad inadecuadas (por ejemplo, métodos estáticos para encriptado). Ello suele combatirse empleando cifrado dinámico. El cifrado de datos dinámico, produce distintos resultados, aunque utilice la misma entrada de texto; esto puede confundir a los ciber-delincuentes. Una propuesta reciente, recomienda utilizar algoritmos genéticos (conocida como: *Noised random pseudo-hexadecimal GAs*). La presente investigación, es continuidad de dicho trabajo, porque se observó que ese algoritmo genético (AG), demora demasiado tiempo y el proceso puede entrar en un ciclo infinito. Esta situación no se considera segura para las organizaciones, aunque el *pseudo-hexadecimal* muestra resultados muy prometedores. Por ello, la importancia de esta investigación, porque se realizan mejoras al procedimiento, omitiendo el uso del AG, siendo reemplazado por un método aleatorio con Inteligencia Artificial (IA). El alcance y objetivo, es reducir tiempo del cifrado de datos y evitar que la ejecución quede detenida en un ciclo infinito. El método propuesto, denominado: *noised random pseudo-hexadecimal*, es recomendado como nueva alternativa en el cifrado dinámico. Finalmente, se comparan resultados de ambas metodologías y se introduce una tabla de equivalencias sobre el formato *pseudo-hexadecimal*.

Palabras claves: *Cifrado de datos dinámico, inteligencia artificial, métodos aleatorios, criptografía, formato Pseudo-Hexadecimal*.

ABSTRACT

The theft of digital data may produce losses in the organisation's finances, if used cybersecurity strategies are not adequate (e.g., employment of static encryption methods). It's been handled with dynamic encryption methodology, which it can create different ciphertext results with the same plain text input. These schemes may help to mistake the cyber-criminals. Recent proposals recommend using genetic algorithms (e.g., Noised random pseudo-hexadecimal GA's). This research is future work of pseudo-hexadecimal GA's strategy because it was observed that genetic algorithm (GA) takes a long time for processing. Besides, it may fall into an infinite cycle. This situation is not secure for data protection of the organisations, despite that pseudo-hexadecimal schema is good alternative. That's why the importance of our work because some modifications for improving the dynamic encryption procedure have been recommended replacing GA by random methodologies with artificial intelligence (AI). Regarding the research's aims, they consist in the reduction of time for data encryption and to avoid that execution processing falls into an infinite cycle. Novel proposal, named here as: Noised Random Pseudo-Hexadecimal, it is recommended as dynamic data encryption alternative. Finally, the comparison of results concerning to both methodologies based on pseudo-hexadecimal are also shown and a table of conversion for hexadecimal to pseudo-hexadecimal encoding is here included.
Keywords: Dynamical encryption methods, artificial intelligence, random methodologies, cryptography, pseudo-hexadecimal formato.

1. INTRODUCCIÓN

El robo digital de datos, es un problema que suele ocurrir en dominios prácticos, cuando alguna de las estrategias de ciberseguridad (Delman, 2004; Kalsi et al., 2018; Mendoza, 2008; Rangel et al. 2023; Reddaiah, 2016, 2019) empleadas son inadecuadas, es decir, al menos uno de los métodos utilizados resulta vulnerable a ataques, por parte de los ciber-delincuentes, situación que puede producir importantes pérdidas en las finanzas de las organizaciones (Rangel & Rangel, 2024). Una de las áreas de estudio que permite resolver este tipo de problemas, se conoce como: criptografía (Barranco & Galindo, 2022; Dang & Le, 2022; Delman, 2004; Granados, 2006; Kalsi et al., 2018; Liu & Wang, 2022; Mendoza, 2008; Reddaiah, 2016, 2019; Rueda, 2005; Singh, 2000), la cual, proporciona métodos, técnicas, algoritmos, estrategias y herramientas que permiten el cifrado de datos. Se entiende por cifrado de datos, a la ocultación de la información, mediante la traducción de un mensaje original convirtiéndolo en un tipo de lenguaje o código, utilizando un alfabeto para cifrado/descifrado, que solamente podrá ser capaz de entender el software especializado o persona autorizada (Rangel et al., 2023). El proceso inverso, se conoce como descifrado de datos (Gómez et al., 2012; Liu & Wang, 2021; Rangel et al., 2023), el cual, consiste en convertir el texto cifrado a texto claro. Comúnmente, estos algoritmos utilizan una llave secreta como parte de la entrada, ya que, de acuerdo con su clasificación, pueden ser considerados: simétricos o asimétricos (Gómez & Díaz, 2021; Mendoza, 2008). La criptografía simétrica, utiliza la misma clave para cifrar y descifrar el mensaje de datos, es decir, se basa en un secreto compartido (Gómez et al., 2012; Rangel et al., 2023). Es por ello, que este tipo de métodos se les conoce como: cifrado de llave simétrica (Gómez & Díaz, 2021; Mendoza, 2008; Rangel et al., 2023; Thakur & Kumar, 2011), debido a que, debe coincidir tanto en el proceso de cifrado como en el de descifrado para que la comunicación entre el emisor quien cifra, como el receptor que descifra, sea exitosa.

Según refiere Singh (2000), la criptografía ha sido usada casi simultáneamente desde el desarrollo avanzado del lenguaje escrito, jugando un rol fundamental en la protección de las comunicaciones oficiales de los Estados, de los gobernantes y, principalmente, de las instituciones militares (Álvarez, 2019). Los conflictos bélicos del siglo XX, fueron los que permitieron el desarrollo de los principales avances en la criptografía moderna con el desarrollo de las teorías de la información de Claude Elwood Shannon, quien es considerado el padre de la criptografía moderna (Granados, 2006; Singh, 2000; Stinson, 2022). En este contexto, Stinson (2022), realiza un análisis al respecto, señalando a William Shakespeare como el primero en utilizar el término "cifrado" en 1597. Sin embargo, considera a Claude Shannon como el padre de la criptografía moderna, debido a su trabajo pionero en la teoría de la información y la criptografía. También, presenta a Alan Turing, como otro pionero en la criptografía moderna, por su contribución al desarrollo de la máquina Enigma durante la Segunda Guerra Mundial (Stinson, 2022). Del mismo modo, en lo que refiere a los algoritmos de cifrado del pasado (Barranco & Galindo, 2022; Gómez et al., 2012; Goyal & Kaur, 2021; Kumar & Sharma, 2021; Rangel et al., 2023; Singh & Kaur, 2021), se encuentra el creado por: Julius Caesar (Barranco & Galindo, 2022; Goyal & Kaur, 2021; Kumar & Sharma, 2021; Luciano & Prichett,1987; Rangel et al.,2023; Rangel & Rangel, 2024; Singh & Kaur, 2021), quien diseñó métodos seguros para transmitir en secreto información para gente importante en el campo de la milicia (Rangel et al., 2023; Reddaiah, 2019; William, 1999). Por otra parte, según Jiménez et al. (2015), es indispensable hacer uso de algún mecanismo que ayude a resguardar la información de ataques maliciosos (Al-Riyami & Al-Khateeb, 2020; Chaudhary & Sharma, 2020; Goyal & Kaur,2020; Patel & Patel, 2020). Al respecto, podemos encontrar en la literatura, que para resolver este tipo de problemas, lo más utilizado en la criptografía son los métodos de cifrado (Gómez & Díaz, 2021; Liu & Wang, 2021; Rangel & Rangel, 2024; Stinson, 2022), cuyo propósito es mantener a salvo la información de las organizaciones, así como de usuarios particulares, que manejan grandes volúmenes de datos (Oppliger, 2005). Estas estrategias de ciberseguridad, sirven para reducir un poco el problema de robo digital de datos, particularmente, si el proceso de encriptación (Gómez & Díaz, 2021; Van-Tilborg H.C.A., 2005; Van and Jajodia, 2011), se lleva a cabo mediante la implementación de nuevos métodos o algoritmos para el cifrado de datos (Rangel et al., 2023). Según Rangel & Rangel (2024), algunas alternativas para amortiguar un poco este problema, consisten en reemplazar periódicamente el algoritmo de encriptado, preferentemente, haciendo uso de métodos de cifrado dinámico. Lo anterior, debido a que las herramientas de desencriptado (Al-Riyami & Al-Khateeb, 2021; Chaudhary & Sharma, 2021; Singh & Kaur, 2021b) de

información que utilizan los ciber-delincuentes suelen estar muy actualizadas. El hecho de no cambiar periódicamente la forma de cifrado de datos en las organizaciones, ello puede poner en riesgo la seguridad de la información. Es por ello, que se considera importante diseñar nuevos algoritmos de cifrado, o en su defecto, realizar modificaciones, a los algoritmos ya existentes, para que los ciber-criminales desconozcan el procedimiento de descifrado, y al menos, tener nuestra información segura, durante un breve lapso de tiempo (Rangel et al., 2023). Según Rangel et al. (2023), esto es, "porque el proceso de cifrado/descifrado... es conocido por los desarrolladores; ya que, varios lenguajes de programación... pueden utilizar paquetes, ... o librerías que permiten el cifrado/descifrado de los algoritmos más populares". Por ejemplo, si un "lenguaje... , permite el cifrado, ... del ... MD5: Message-Digest Algorithm 5 (Dhany et al., 2018; Freda, 2022; Kumar & Singh, 2022), ... bastaría «entrenar un modelo supervisado» durante un ... tiempo, para construir un diccionario... mediante procesamiento de lenguaje natural ... , para poder descifrar, ... un texto ...". Derivado de esta situación, algunos autores han realizado investigaciones, introduciendo nuevas propuestas, que pueden ayudarnos a reemplazar periódicamente los métodos de cifrado de datos en las organizaciones, ya que, según Rangel et al. (2023), "es recomendable proteger la información utilizando algoritmos nuevos, los cuales, no incluyan soporte los lenguajes de programación libres o comerciales. Dicha situación, retrasa o demora a los hackers apoderarse de nuestra información".

Si observamos en la literatura, podemos encontrar un gran número de propuestas para el cifrado de datos (Van-Tilborg H.C.A., 2005), las cuales, siguen diferentes caminos en su desarrollo. Un caso particular, son los métodos y algoritmos asimétricos y simétricos (Gómez & Díaz, 2021; Mendoza, 2008), basados en alfabetos que permiten el encriptado de datos usando cifrados de flujo, por permutación, por desplazamiento y/o sustitución de carácteres (Barranco & Galindo, 2022; Luciano and Prichett, 1987; Anónimo: The Dojo, 2021). En este contexto, el trabajo presentado por Gómez et al. (2012), muestra algunas técnicas de encriptación clásica, básicas y modernas, realizando un cripto-análisis (Dang & Le, 2022; Liu & Wang, 2022; Reddaiah, 2016), sobre métodos clásicos de cifrado, basadas en la teoría de la información y la estadística, argumentando que pueden ser usadas en otros métodos de cifrado, si las mismas debilidades están presentes, por esta razón, según Gómez et al. (2012), es útil conocer estas técnicas cuando se diseñan nuevos algoritmos. Ejemplos de estos algoritmos, son el cifrado del César (Barranco & Galindo, 2022; Gómez et al., 2012; Goyal & Kaur, 2021; Kumar & Sharma, 2021; Singh & Kaur, 2021), el cual, fue muy popular en sus inicios, por ser sencillo de implementar (Rangel et al., 2023). Se trata de un algoritmo de desplazamiento o sustitución, también conocido como: algoritmo Caesar (Barranco & Galindo, 2022; Gómez et al., 2012; Goyal & Kaur, 2021; Kumar & Sharma, 2021). Este algoritmo, tiene algunas variantes, como es el caso del cifrado Vigenère (Gómez et al., 2012; Goyal & Kaur, 2021; Singh & Kaur, 2021). La seguridad proporcionada por este tipo de cripto-sistemas se basa completamente en la construcción de mecanismos con una gran variedad de funciones y operaciones (Reddaiah, 2016). Empero, los cifrados basados en Caesar, ya no son muy utilizados en la actualidad (Rangel et al., 2023), debido a que, han destacado otras propuestas (Progress Software Corporation Telerik, 2022), que proporcionan mayor seguridad de la información; tal es el caso de los algoritmos que realizan el cifrado de datos mediante utilización de claves privadas (Rahman & Hossain, 2021; Saini & Gupta, 2021; Thakur & Kumar, 2011) o encriptado usando claves públicas (Liu & Chen, 2021; Luciano & Prichett, 1987; Wang & Zhang, 2021); así como, los que permiten encriptado basado en funciones hash (Freda, 2022; KeepCoding, 2023; Lake, 2023; Lowery, 2023; Mendoza, 2008; PortalCripto, 2023; Progress Software Corporation Telerik, 2022), por mencionar algunos.

Con respecto a los algoritmos de cifrado basados en claves públicas, uno de los estándares muy popular es conocido como: RSA, por sus siglas: "Rivest, Shamir and Adleman" (Dang & Le, 2022; Luciano & Prichett, 1987; Mendoza, 2008; Rahman & Hossain, 2021; Rodríguez, 2020). En lo que concierne con los algoritmos de cifrado basados en claves privadas, dos de los más populares son conocidos como: DES (Dhany et al., 2018; Kumar & Sharma, 2021) y AES (Rahman & Hossain, 2021; Rodríguez, 2020; Singh & Kumar, 2022a, 2022b), por sus siglas en inglés: "Data Encryption Standard" y "Advanced Encryption Standard", respectivamente (Dhany et al., 2018; Kumar & Sharma, 2021; Rahman & Hossain, 2021; Rodríguez, 2020; Thakur & Kumar, 2011; Singh & Kumar, 2022a, 2022b). También, existen las propuestas relacionadas con el cifrado de datos basados en funciones hash

(Álvarez, 2019; Courtois, 2012; Dhany et al., 2018; Freda, 2022; Fulgueira, et al., 2015; Gómez, et al., 2012; KeepCoding, 2023; Lake, 2023; Lowery, 2023; Lowery & Pereyra, 2023; Nagaraj & Srinadth, 2015; Pieprzyk & Tombak,1994; PortalCripto, 2023; Progress Software Corporation Telerik, 2022; Singh & Kumar, 2022a, 2022b) y derivados. Un ejemplo de este tipo de encriptado, es el algoritmo conocido como: MD5, por sus siglas en inglés: Message-Digest Algorithm 5 (Dhany et al., 2018; Freda, 2022; Gómez et al., 2012; Kumar & Singh, 2022; Lake, 2023; Lowery, 2023; Lowery & Pereyra, 2023). Otro ejemplo, es el algoritmo SHA-1, por sus siglas en inglés: Secure Hash Algorithm 1 (Gómez et al., 2012; KeepCoding, 2023; Kumar & Singh, 2022; Li & Wang, 2022; Lowery, 2023; Lowery & Pereyra, 2023; PortalCripto, 2023). Finalmente, un caso de encriptado derivado de funciones hash, es el método GOST, por su significado: GOsudarstvennyi STandard (Álvarez, 2019; Courtois, 2012; Fulgueira et al., 2015; Pieprzyk & Tombak, 1994; Singh & Kumar, 2022b). Otra alternativa para el cifrado de datos, además del basado en funciones hash, también existe el encriptado utilizando tecnología óptica (Goodman, 1968; Javidi & Horner, 1994; Javidi et al., 1997; Linfei & Daomu, 2005; Mogensen & Glückstad, 2000a, 2000b, 2001a, 2001b; Nomura & Javidi, 2000; Rueda & Lasprilla, 2002; Rueda et al., 2005; Singh & Kumar, 2022a, 2022b; Tajahuerce & Javidi, 2000a; Tajahuerce et al., 2000b; Tajahuerce et al., 2001a; Tan et al., 2000; Wang & Chen, 2022), así como, otro paradigma o esquema de comunicación segura desarrollado para cifrar información, el cual, utiliza los sistemas discretos caóticos (Chong et al., 2011; Hossam et al., 2007; Jiménez et al., 2015; Liu & Zhang, 2021; Pisarchik & Flores-Carmona, 2006; Pisarchik & Zanin, 2008; Rajan & Saumitr, 2006; Wang & Chen, 2021), dichos procedimientos, pueden utilizarse para el cifrado de datos dinámico (Rangel & Rangel, 2024).

Además de los cripto-sistemas basados en métodos o algoritmos simétricos y asimétricos, los que utilizan claves públicas y privadas, los basados en funciones hash, los trabajan con tecnología óptica, los que usan sistemas discretos caóticos; también existen los métodos o algoritmos de cifrado que hacen uso de la inteligencia artificial (Iyengar et al., 2021a, 2021b; Rangel & Rangel, 2024; Singh, et al., 2021), que según Rangel (2002, 2022), esta área de estudio pertenece al campo de las Ciencias Computacionales, dónde uno de los principales propósitos es hacer emular (o simular) que "la máquina piense". Para conseguir dicho propósito, en la inteligencia artificial (IA) existen diferentes alternativas, modelos, estrategias, arquitecturas, paradigmas, técnicas y métodos. Cada uno de ellos, sustentado a través de distintas vertientes. Por ejemplo, algunos tienen sus bases en métodos heurísticos, debido a que consisten en un conjunto de reglas definidas para resolver determinado problema, como el caso de gráfos (Hartmann, 2020; Russell & Norvig, 2020; Sánchez et al., 1997b) y árboles de decisión (Mitchell, 2020; Ross-Quinlan, 1993; Russell & Norvig, 2020). También, existen modelos que se apoyan en métodos estadísticos (Bishop, 2022; Kanal, 1963) y métodos aleatorios (Kuncheva & Jain, 1999; Murphy, 2022; Reddaiah, 2016; Skalak, 1994), los cuales, consisten en la generación de números al azar que pueden obtenerse sin reemplazo (sin repetición) o con repetición (con reemplazo); entre otros paradigmas que se han aplicado en distintos campos de estudio, como es el caso de robótica móvil (Arai & Sato, 2021; Rangel & García, 2002; Siegwart & Nourbakhsh, 2011), la criptografía basada en IA (Dang & Le, 2022; Delman, 2004; Kalsi et. al., 2018; Liu & Wang, 2022; Mendoza, 2008; Rangel & Rangel, 2024; Reddaiah, 2019; Sebas, 2023), el reconocimiento de formas conocido también como: reconocimiento de patrones (Bishop, 2022; Johns, 1961; Rangel, 2002; Sebestyen, 1962), las redes neuronales artificiales (Bruzzone & Serpico, 1997; Goodfellow et al., 2021; Murphy, 2022), la minería de datos (Han et al., 2022; Hand et al., 2001; Tan et al., 2022; Zhang & Yang, 2022), el aprendizaje automático, conocido como: machine learning (Chen & Wang, 2022; Goodfellow et al., 2021; Murphy, 2022; Nils-Nilson, 1965), que en su nueva actualización se le conoce como: aprendizaje profundo o deep learning (Goodfellow et al., 2016; Krizhevsky et al., 2012; LeCun et al., 2015; Zhang & Yang, 2022); incluyendo otros modelos basados en métodos aleatorios, como es el caso de los algoritmos genéticos (Clark, 1994; Delman, 2004; Griindlingh & Van-Vuuren, 2002; Kalsi et al., 2018; Kuncheva & Jain, 1999; Matthews, 1993; Rangel et al., 2023; Reddaiah, 2019; Sebas, 2023; Skalak,1994), por mencionar algunas áreas relacionadas.

Los algoritmos genéticos (AG en español, o GA, por sus siglas en inglés: Genetic Algorithms), también han sido utilizados para el cifrado de datos. Esta alternativa, no es nueva, ya que, se han empleado en varias investigaciones mediante el uso de distintas variantes (Delman, 2004; Kalsi et. al., 2018; Liu & Chen, 2022; Rangel et al., 2023;

Rangel & Rangel, 2024; Reddaiah, 2019; Rodríguez, 2020), incluso aplicado al área de cripto-análisis (Barker & Roginsky, 2020; Dang & Le, 2022; Gómez et al., 2012; Katz.& Lindell, 2019; Liu & Wang, 2022; Nagaraj & Srinadth, 2015). Un estudio relacionado fué propuesto por Delman (2004), quien presenta el uso de GA aplicado al área de criptografía, exponiendo dos propuestas: la primera relacionada con cripto-análisis y otra utilizando una metodología basada en GA. La comparación de resultados, revelan retardo de tiempos y porcentaje de descrifrado satisfactorios. Según Delman (2004), otro trabajo al respecto, está relacionado con ataques sobre la transposición y permutación de cifrado, en algunos casos, aplicado a mono-alfabetos, que entre ellos destacan: Clark (1994), Griindlingh & Van-Vuuren (2002) y Matthews (1993). La aportación de Matthews (1993), es respecto al uso de GA en tareas de cripto-análisis; mientras que Clark (1994), propone una moderna optimización de algoritmos para el cripto-análisis. Del mismo modo, Griindlingh & Van-Vuuren, 2002), utilizan GA para romper la seguridad de un simple cifrado criptográfico. También, Reddaiah (2016), después de haber llevado a cabo una investigación concerniente con funciones de emparejamiento aplicado a la criptografía, realiza un estudio sobre algoritmos genéticos en éste mismo campo, que mediante inteligencia artificial, aportando al área de comercio electrónico (Reddaiah, 2019). En este trabajo, Reddaiah (2019), propone GA aplicado a la criptografía en el campo de e-commerce (comercio electrónico), presentando funciones robustas en el procesamiento, usando diferentes tipos de GA, para proporcionar seguridad de los datos. En dicho trabajo, se discuten las ventajas y desventajas de las funciones basadas en GA presentadas, concluyendo que esta tendencia es de gran importancia para la seguridad; y, aunque el tamaño de la clave es reducido por GAs, se reporta buena seguridad de los datos. En cambio, Rodríguez (2020), trabaja con operadores genéticos aplicados a la criptografía simétrica aplicando GA, entropía y aritmética modular; mostrando comparación de resultados con los algoritmos: DES (Data Encryption Standard), RSA (Rivest, Shamir and Adleman) y AES (Advanced Encryption Standard). En esta investigación, Rodríguez (2020) propone el algoritmo criptográfico simétrico para texto, empleando una metodología experimental dentro de un sistema determinista, redistribuyendo y modificando los parámetros y fases del GA, que afectan directamente el comportamiento, logrando un cifrado independiente para la clave auxiliar, a través del uso de una clave principal, encargada de aumentar la seguridad, exponiendo factores como: escalabilidad, tamaño de la clave y tiempo de ejecución, señalando que el algoritmo propuesto tiene un buen desempeño en estos términos (Rodríguez, 2020). Adicionalmente, Liu & Chen (2022), exponen resultados sobre un estudio de cripto-análisis basado en funciones hash del algoritmo MD5, pero lo realizan empleando algoritmos genéticos.

Sin embargo, un reciente estudio relacionado fue realizado por Rangel et al. (2023), en el cual, se propone un algoritmo genético para el cifrado de datos utilizando un nuevo concepto denominado formato *pseudo-hexadecimal*, con el propósito de inyectar ruido a los mensajes cifrados. Esta propuesta, que recomienda utilizar GA, presenta una alternativa denominada: *Noised random pseudo-hexadecimal GAs* (*algoritmo genético aleatorio con ruido pseudo-hexadecimal*). La presente investigación, es continuidad de dicho trabajo (Rangel et al., 2023), porque se observó que ese algoritmo genético "ruidoso", demora demasiado tiempo en alcanzar la convergencia y el proceso puede entrar en un ciclo infinito. Esta situación no se considera segura para las organizaciones, aunque el *pseudo-hexadecimal* muestra resultados muy prometedores. Es por ello, la importancia de esta investigación, porque se realizan mejoras al procedimiento de Rangel et al. (2023), omitiendo el uso del GA, siendo reemplazado por un método aleatorio con inteligencia artificial (IA).

Además, se considera importante la presente investigación, para poder dar aportación empírica a favor del denominado: *pseudo-hexadecimal,* debido a que, dicho formato no es considerado una codificación dentro de los estándares internacionales, y por ende, ha sido poco estudiado en la literatura. Las primeras investigaciones realizadas respecto al uso de dicho formato, no fueron publicadas formalmente, solo mediante conferencias o ponencias nacionales, o en forma institucional. Por lo tanto, algunos editores de revistas han rechazado este tipo de propuestas porque no constituyen una base formal dentro del campo científico; mientras que otras investigaciones (Rangel & Rangel, en revisión desde 2024) donde se insiste sobre el uso o aplicación del *pseudo-hexadecimal*, todavía se encuentran en proceso de revisión por parte de algunas editoriales de revistas científicas y una publicación al respecto (Rangel et al., 2023), aún se encuentra en trámite el registro del ISSN por parte del editor. Lo anterior constituye una razón, sobre la justificación del presente trabajo de investigación: "para poder dar

aportación empírica a favor del uso de este tipo formato *pseudo-hexadecimal*", que se presume permite camuflajear el contenido de un texto cifrado, haciendo difícil su desencriptado a los ciber-delincuentes, debido a que, el contenido del mensaje cifrado, para una misma cadena de texto, difiere en cada ejecución del algoritmo, porque se tiene inyectado ruido en el mensaje cifrado, y al utilizar el concepto: pseudo-hexadecimal, resulta menos vulnerable a ataques (Rangel et al., 2023). En este mismo trabajo señala que: "al momento, no se ha podido descifrar el paquete generado por: Noised pseudo-hexadecimal GAs, ya que, dichas herramientas... especializadas para el descifrado de datos, solicitan información, ... entre otros datos; los cuales no contiene el paquete generado por la... propuesta... basada en *pseudo-hexadecimal*".

Una de las primeras aplicaciones del denominado: *pseudo-hexadecimal,* que lograron ser publicadas (Rangel et al., 2023), fue inyectando ruido en el cifrado de datos usando algoritmos genéticos, siendo empleado como alfabeto secundario, mezclado con números hexadecimales de un previo alfabeto primario, convirtiendo en pseudo-hexadecimal al texto cifrado. Lo anterior, llevaba como propósito confundir a los ciber-delincuentes, al momento de querer descifrar los datos. Este estudio reciente que concerniente con el uso del formato *pseudo-hexadecimal*, fué presentado por Rangel et al. (2023), en el cual, se comenta que "lleva como propósito inyectar ruido al paquete de cifrado generado", refiriendo que ese "trabajo es un reporte técnico de la investigación, debido a que, al momento solamente se presentan resultados preliminares, puesto que, falta concluir algunos experimentos...". Otro comentario sobre el "nuevo concepto denominado: *pseudo-hexadecimal*", señala en dicha fuente de referencia, que permite "impedir que software especializado para descrifrado de datos traduzca la información cifrada, o en su defecto, pueda demorar o retardar un poco el descifrado de datos". Otro aspecto mencionado en dicha investigación, señala que el *pseudo-hexadecimal* trabaja con dos cifras (por ejemplo: FF), empleando notación del formato muy parecido al hexadecimal, pero no lo es (Rangel et al., 2023). También, declara que el *pseudo-hexadecimal* "consiste en introducir ruido a algunos números hexadecimales, generando secuencias con otros caracteres del ASCII, parecidos a hexadecimal, pero no se encuentran en un rango válido... Por ejemplo: el hexadecimal del caracter N es 4e y se escribe como 0x4E ; y para inyectar ruido el valor .... sería: 0x0N, lo cual, en números hexadecimales no es válido, todo ello con la finalidad de conseguir que el texto cifrado sea descifrado con mayor dificultad... agregando ruido en formato: *pseudo-hexadecimal*, simulando los caracteres del ASCII como número hexadecimal...." (Rangel et al., 2023). En esa misma fuente (Rangel et al., 2023), se presenta un ejemplo, teniendo entendido que el vector: C' [i] = [B , U , E , ^ , X , @ , N , Q , A , O], es un alfabeto secundario, utilizado para el cifrado de datos. Por lo tanto, su correspondiente *pseudo-hexadecimal* quedaría como se muestra enseguida: PseudoHexa(C' [i]) = [0x0B , 0x0U , 0x0E , 0x0^ , 0x0X , 0x0@ , 0x0N , 0x0Q , 0x0A , 0x0O]; mientras que su equivalente hexadecimal sería: Hex (C' [i]) = [42 , 55 , 45 , 5e , 58 , 40 , 4e, 51 , 41 , 4f]. Sin embargo, no se aclara la razón del porqué el caracter ASCII: "B" con hexadecimal: 42, se convierte a *pseudo-hexadecimal* como: "0B". Del mismo modo, no se específica porqué el caracter ASCII: "N", su *pseudo-hexadecimal* es: "0N"; y tampoco se presenta alguna tabla de equivalencias: ASCII versus *pseudo-hexadecimal*.

Empero, podemos observar en otro trabajo de investigación (Rangel & Rangel, en revisión desde 2024), que se trata de aclarar un poco esta situación, al momento que se presenta una fórmula o planteamiento matemático para obtención del denominado *pseudo-hexadecimal*. En dicha referencia, se señala que "la inyección de ruido en el texto cifrado con formato *pseudo-hexadecimal*, consiste en almacenar números que son muy parecidos a la codificación hexadecimal, que incluye otros caracteres ASCII que están condicionados...". La condición para incluir este tipo de información, puede ser calculada por la siguiente ecuación: $PSH_i = PseudoHexadecimal_i = ("0" + S_i)$. Donde: El operador + refiere a la función de concatenación y el vector S_i es el texto plano en formato ASCII. En este mismo contexto, Rangel & Rangel (en revisión), hacen la siguiente aclaración sobre porqué este formato *pseudo-hexadecimal* permite camuflajear algunos caracteres, refiriéndose como se describe enseguida: "la vocal «A» del ASCII, su número ordinal (entero) es 65, mientras que su hexadecimal es 41 y su correspondiente *pseudo-hexadecimal* es: 0A". Este valor está camuflajeado, debido a que, el valor 0A visto como hexadecimal corresponde en la tabla ASCII al ordinal 10, el cual, refiere a una secuencia de escape de nueva línea. En esos términos, el caracter ASCII: «A», se dice está camuflajeado en *pseudo-hexadecimal* como: 0A (es decir, caracter de nueva línea, en hexadecimal convertido a ordinal ASCII). Por lo tanto, el caracter «A», está oculto en el texto

cifrado como un caracter de nueva línea. Por lo anterior, en lo que concierne con la aportación de la presente investigación, se introduce una tabla de equivalencias: ASCII, ordinal, hexadecimal, *pseudo-hexadecimal* (ver Tabla 1).

Dentro de este mismo contexto, de acuerdo con los referentes sobre el algoritmo genético aleatorio con ruido pseudo-hexadecimal, un primer método de encriptado de datos publicado (Rangel et al., 2023), el cual, fue empleado en la fase de experimentación de la presente investigación, es el denominado: *Noised random pseudo-hexadecimal GAs*. Este algoritmo, fue presentado por primera vez con el propósito de confundir a los ciber-delincuentes (Rangel et al., 2023), inyectando ruido a los mensajes cifrados utilizando este nuevo formato denominado: *pseudo-hexadecimal*, bajo la premisa de ser desconocido por los ciber-criminales. Tal vez, sea la razón, por la cual, no se describen muchos detalles sobre este tipo de formato *pseudo-hexadecimal*, para evitar la temprana interpretación de dicha codificación. En la referencia (Rangel et al., 2023), no específica claramente si el método: *Noised random pseudo-hexadecimal GAs*, se trata de un tipo de encriptado por sustitución o desplazamiento, debido a que, se plantean ambas estrategias en la descripción del algoritmo. Ello da lugar, a realizar varios tipos de implementación o modificaciones al respecto. Sin embargo, en las especificaciones proporcionadas en dicha fuente, para el cifrado de datos se describe como opcional este esquema, durante la primera fase o etapa. En primer lugar, se considera como primer opción el uso de un algoritmo genético, denominado: *noised cypher GAs dictionary*: *algoritmo genético para generar alfabeto con cifrado ruidoso* (Rangel et al., 2023), que permitirá la generación de dos alfabetos: uno primario (para descrifrado) y otro secundario (para encriptado). En estos términos, se entiende que el cifrado podría realizarse por sustitución, ya que, al ser generado aleatoriamente, se intuye irrelevante la realización de un desplazamiento, porque podría tener lugar el mismo resultado, después de una *n-ésima* ejecución posterior al método aleatorio. Sin embargo, Rangel et al. (2023), describe en la segunda parte del algoritmo, la existencia de un vector K_i de desplazamientos. Ello se entiende como una segunda opción para generar el alfabeto secundario, y para realizar el cifrado de datos.

Tabla 1: Equivalencias ASCII (A), ordinal (O), hexadecimal (H), pseudo-hexadecimal (Psh), utilizando módulo 120 (mod 120, comprende el rango ASCII desde 30 hasta 150). El caracter "" del ordinal 30, no es imprimible en pantalla, el cual, se parece a otros caracteres no imprimibles en pantalla dentro de esta misma tabla, por ejemplo: "" del ordinal 137. Sin embargo, son caracteres ASCII diferentes.

A	O	H	Psh	A	O	H	Psh	A	O	H	Psh	A	O	H	Psh
	30	1e	0	=	61	3d	0=	\	92	5c	0\	{	123	7b	0{
	31	1f	0	>	62	3e	0>	]	93	5d	0]	|	124	7c	0|
(sp)	32	20	0(sp)	?	63	3f	0?	^	94	5e	0^	}	125	7d	0}
!	33	21	0!	@	64	40	0@	_	95	5f	0_	~	126	7e	0~
"	34	22	0"	A	65	41	0A	`	96	60	0`		127	7f	0
#	35	23	0#	B	66	42	0B	a	97	61	0a		128	80	0
$	36	24	0$	C	67	43	0C	b	98	62	0b		129	81	0
%	37	25	0%	D	68	44	0D	c	99	63	0c		130	82	0
&	38	26	0&	E	69	45	0E	d	100	64	0d		131	83	0
'	39	27	0'	F	70	46	0F	e	101	65	0e		132	84	0
(	40	28	0(	G	71	47	0G	f	102	66	0f		133	85	0
)	41	29	0)	H	72	48	0H	g	103	67	0g		134	86	0
*	42	2a	0*	I	73	49	0I	h	104	68	0h		135	87	0
+	43	2b	0+	J	74	4a	0J	i	105	69	0i		136	88	0

,	44	2c	0,	K	75	4b	0K	j	106	6a	0j	137	89	0
-	45	2d	0-	L	76	4c	0L	k	107	6b	0k	138	8a	0
.	46	2e	0.	M	77	4d	0M	l	108	6c	0l	139	8b	0
/	47	2f	0/	N	78	4e	0N	m	109	6d	0m	140	8c	0
0	48	30	00	O	79	4f	0O	n	110	6e	0n	141	8d	0
1	49	31	01	P	80	50	0P	o	111	6f	0o	142	8e	0
2	50	32	02	Q	81	51	0Q	p	112	70	0p	143	8f	0
3	51	33	03	R	82	52	0R	q	113	71	0q	144	90	0
4	52	34	04	S	83	53	0S	r	114	72	0r	145	91	0
5	53	35	05	T	84	54	0T	s	115	73	0s	146	92	0
6	54	36	06	U	85	55	0U	t	116	74	0t	147	93	0
7	55	37	07	V	86	56	0V	u	117	75	0u	148	94	0
8	56	38	08	W	87	57	0W	v	118	76	0v	149	95	0
9	57	39	09	X	88	58	0X	w	119	77	0w	150	96	0
:	58	3a	0:	Y	89	59	0Y	x	120	78	0x			
;	59	3b	0;	Z	90	5a	0Z	y	121	79	0y			
<	60	3c	0<	[	91	5b	0[	z	122	7a	0z			

Lo anterior, también podría ser innecesario, porque al ser aleatorios los valores de los vectores que guardan los alfabetos de cifrado/descrifrado, hacer un desplazamiento podría incurrir en algún momento, en el mismo resultado que empleando sustitución, esto es, después de haber ejecutado cierto número de veces el procedimiento aleatorio que permite generar los alfabetos. En cambio, para el descrifrado de datos, Rangel et al. (2023), describen un procedimiento basado en sustitución. En términos generales, según refiere Rangel et al. (2023), el denominado: *Noised random pseudo-hexadecimal GAs*, consiste en tres etapas o fases.

La *primera parte o fase*, del *Noised random pseudo-hexadecimal GAs*, consiste en utilizar el algoritmo genético denominado: *noised cypher GAs dictionary* (*algoritmo genético para generar alfabeto con cifrado ruidoso*). La ejecución de este procedimiento, permite generar los alfabetos primario y secundario, para descrifrado/cifrado de datos, respectivamente; garantizando un eficiente grado de aleatoriedad sin reemplazo o coincidencias. Este método, por tratarse de un algoritmo genético, se compone al menos de cuatro fases: selección, cruce, mutación y evaluación (Rangel et al., 2023). El referido: *noised cypher GAs dictionary*, inicia su procedimiento de selección, utilizando métodos aleatorios con reemplazo, con el propósito de generar un vector inicial, que representará el alfabeto primario con un módulo 120 (mod 120), porque utiliza solamente 120 caracteres. Ello quiere decir, que los caracteres ASCII permitidos están dentro del rango ordinal con valores entre 30 y 150. En la investigación presentada por Rangel et al. (2023), se ha denominado a este alfabeto primario como: vector ABC, el cual, se describe guarda sus valores en formato hexadecimal. Posteriormente, se procede a crear una muestra de entrenamiento de dos clases. Una clase representa los vectores más parecidos al alfabeto primario (clase=1, los vecinos más cercanos) y la segunda clase representa a los vectores menos parecidos al denominado: ABC (clase=2, los vecinos más lejanos); excepto en la primera generación del algoritmo genético, porque todos los ejemplares son de clase=0. Cada ejemplar de la muestra de entrenamiento, simula ser un cromosoma de bytes (porque cada valor ASCII, es un caracter o byte). El número de columnas de cada cromosoma, es del mismo tamaño del alfabeto primario (es decir, 120 posiciones por la extensión del mod 120), agregando una columna adicional para guardar la etiqueta de clase. Estos cien cromosomas almacenados en la muestra de entrenamiento, son generados de manera aleatoria con reemplazo y cada uno de ellos representa una posible solución, que en este caso, se trata de un alfabeto que servirá para el descrifrado de datos. El mod 120, inicia en el ordinal ASCII 30 y

termina en 150. Por lo tanto, existen varios caracteres no imprimibles en pantalla, lo cual, según Rangel et al. (2023), se hace intencionalmente para incorporar ruido a la ME. De lo contrario, se recomienda utilizar un mod 94 (que inicia en el ordinal ASCII 32 y finaliza en el 126), para que todos los caracteres seleccionados puedan ser imprimibles en pantalla. Una vez que se ha creado la muestra de entrenamiento (ME) por primera vez, todos los cromosomas son de clase=0. Entonces, se procede a seleccionar de manera secuencial cada par de cromosomas hasta agotar los ejemplares almacenados en la ME. Cada vez que se selecciona un par de ejemplares, se calcula la distancia euclideana (Rangel, 2002, 2022) entre el alfabeto primario ABC y dichos cromosomas. Se asigna etiqueta de clase=1, al ejemplar que obtuvo la distancia más pequeña y se asigna clase=2, al cromosoma que obtuvo la distancia mayor. Estos valores se actualizan en la ME. Al finalizar, debe actualizar el vector: ABC, asignando los valores de un cromosoma de clase=1, el cual, es seleccionado de manera aleatoria. También, deberá seleccionarse aleatoriamente un cromosoma de clase=2, para ser guardado como alfabeto secundario. En la referencia Rangel et al. (2023), es denominado como: xABC, el cual, representa el alfabeto para cifrado de datos datos. Este último alfabeto, al concluir todo el procedimiento, se guardan sus valores con formato pseudo-hexadecimal. Con respecto a la distancia euclideana, puede calcularse del siguiente modo: $DE\ (X,Y) = SQRT\ (\ SUM\ (\ X_i — Y_i\ )^{\ 2}\ )$. Donde: *DE* es la distancia euclideana, X_i representa el alfabeto primario, mientras que Y_i es un ejemplar de la ME. La función *SUM* indica sumatoria de ($(X_i - Y_i)$ * $(X_i - Y_i)$) que equivale la representación: $(X_i — Y_i)^2$; y, la función *SQRT* es el cálculo final de la raíz cuadrada. Para poder dar lugar, a una siguiente generación del algoritmo genético, se procede a eliminar de manera aleatoria, el 50% de cromosomas de ambas clases (se eliminan: 25 ejemplares de la ME de clase=1 y 25 cromosomas de clase=2). Ahora, se procede a aplicar la operación cruce, de manera secuencial, seleccionando un par de cromosomas de clase distinta (un ejemplar de clase=1 y un ejemplar de clase=2), hasta agotar los elementos en la ME. Para cada par de cromosomas, se decide aleatoria el método de cruce a emplear. Solo se permite cruce por la mitad y cruce en dos puntos, es decir, por los extremos (Rangel et al., 2023). Después de cruzar cada par de cromosomas, se obtendrán dos hijos o descendientes. Dichos ejemplares se almacenan en la ME con etiqueta de clase=0. Posteriormente, se aplica el proceso de mutación, no solamente a los ejemplares de la ME, sino también a los alfabetos primario y secundario (los denominados: ABC y xABC). El procedimiento de mutación a emplear, también se decide aleatoriamente. Es permitido únicamente realizar la mutación por sustitución o intercambio, y solamente se aplica al 10% o 20% de cada grupo de clases, almacenados en la ME (Rangel et al., 2023). Para el primer caso, se selecciona aleatoriamente la localidad del vector (cromosoma) que desea mutar, y se cambia su valor de manera arbitraria, por cualquier otro elegido de manera aleatoria. En caso del procedimiento por intercambio, se seleccionan aleatoriamente dos localidades del vector (cromosoma) y se intercambian sus valores. Todo el procedimiento anterior, se repite en forma iterativa, hasta haber concluido un máximo de 100 generaciones. Entonces, se procede a evaluar utilizando la siguiente función de aptitud: $P = (\ (\ SUM(\ SIM(ABC_i, xABC_i)\ )\ *\ 100\ )\ /\ m\ )$. Donde: El parámetro *m* es el número de columnas o módulo (por ejemplo, *m=120* para *mod 120*). La función *SIM*, es una métrica de similaridad, que permite evaluar el grado de similitud que existe entre los alfabetos: ABC y xABC. Dicha función, retorna 1, si el valor de la localidad *i* del vector ABC es igual que $xABC_i$; si son diferentes regresa valor de 0. La función *SUM* es la sumatoria de todas las comparaciones. Si el resultado es igual al número del módulo, se entiende que los alfabetos aún no son aptos, porque podrían ser iguales. El resultado de *P*, indica el porcentaje de similitud que tienen los alfabetos: ABC y xABC. El valor ideal para terminar con el procedimiento del algoritmo genético, es un valor de *P=0*, en caso contrario, se repite nuevamente todo el procedimiento (selección, cruce, mutación y evaluación), otras 100 generaciones. Si tarda demasiado tiempo en alcanzar la convergencia (es decir, *P=0*), deberá forzarse la salida del procedimiento, anotando en qué generación ocurrió ello. Por lo tanto, no es obligatorio exigir la convergencia de *P=0*, se puede permitir la repetición de valores en las localidades, con una convergencia de *P* en un rango entre 5 y 10 coincidencias. También, no es obligatorio realizar 100 generaciones en cada proceso de evaluación o iteración. Por lo tanto, el número de generaciones puede ser definido de manera anticipada (Rangel et al., 2023) por el usuario o cripto-analista. Al finalizar, tendremos como resultado un vector: ABC, que representa el alfabeto primario; así como, el alfabeto secundario denominado: xABC. En lo que concierne con el cifrado de datos basado exclusivamente en el algoritmo genético: *noised cypher GAs dictionary* (Rangel et al., 2023), se consigue utilizando el mecanismo por sustitución. Es decir, para una entrada de texto plano: S_i, se cifra buscando una ocurrencia en el alfabeto primario: ABC y se sustituye por el valor correspondiente localizado en el alfabeto secundario: xABC. Al final, se tiene como texto cifrado: C_i, la

concatenación de caracteres encontrados en: xABC, para cada ocurrencia en: S_i. En caso del algoritmo: *Noised random pseudo-hexadecimal GAs*, no es obligatorio utilizar ambos alfabetos: ABC y xABC para el cifrado de datos, ya que, solamente puede seleccionar uno de ellos como alfabeto primario, para comenzar el proceso de desplazamiento siendo controlado por un vector denominado: K_i, lo cual, permite la generación de un nuevo vector, que contiene el alfabeto para cifrado de datos. Sin embargo, según Rangel et al. (2023), si queremos evitar el cálculo de los desplazamientos K_i, podemos hacer uso de los vectores: ABC y xABC, pero a este último vector, tendremos que aplicarle el concepto: *pseudo-hexadecimal*, para generar un cifrado parcial en dicho formato, y poder saltar a la *fase o parte 3*, del algoritmo: *Noised random pseudo-hexadecimal GAs.*

La *segunda parte o fase*, del método: *Noised random pseudo-hexadecimal GAs*, corresponde al cifrado parcial (Rangel et al., 2023). Se trata de ocultar texto, utilizando el alfabeto secundario para cifrado de datos. Esta es la parte, donde se plantea un proceso opcional (por sustitución y/o desplazamiento). El primer caso, es más simple, porque consiste en utilizar el alfabeto secundario (xABC), generado por el algoritmo genético: *noised cypher GAs dictionary*. Posteriormente, debe convertirse el vector: xABC, en formato: *pseudo-hexadecimal*. Esta condición intuye que el cifrado parcial se realizará por mecanismo de sustitución, el cual, ya se ha comentado anticipadamente. Una segunda alternativa para obtener el alfabeto secundario: xABC, que permitirá el cifrado parcial, es emplear un procedimiento de desplazamiento: K_i. Se trata de un vector, seleccionado aleatoriamente con reemplazo, el cual, permite aplicar un desplazamiento distinto a cada caracter del texto plano de entrada. Al respecto, Rangel et al. (2023), explica lo siguiente: Suponiendo que el alfabeto primario es el vector: C_i = [Q , A , N , E , U , @ , ^ , B , O , X] y su correspondiente vector de desplazamiento, seleccionado aleatoriamente con reemplazo, es: K_i = [7 , 3 , 1 , 3 , 5 , 0 , 6 , 3 , 1 , 0]. Al ser aplicado el desplazamiento K_i sobre C_i, teniendo en cuenta que las posiciones del vector empiezan en 1, obtenemos el vector de sustitución para cifrado: C'_i = [B , U , E , ^ , X , @ , N , Q , A , O]; lo anterior, porque el caracter Q se encuentra en la posición 1 de C_i (es decir: C_1) y al sumar su desplazamiento K_i con valor de 7, entonces: 1+7=8, y el caracter ubicado en la posición C_8 es B; en cambio, el caracter *A* se encuentra en la posición 2 de C_i (es decir: C_2) y al sumar su desplazamiento K_i con valor de 3, entonces: 2+3=5, y el caracter ubicado en la posición C_5 es U; y así sucesivamente. Para finalizar, esta segunda parte o etapa del algoritmo, deberá verificar que el alfabeto C_i (o ABC) guarda sus valores en codificación hexadecimal. Del mismo modo, debe convertirse el vector C'_i (o xABC, según corresponda), al nuevo formato denominado como: *pseudo-hexadecimal*. Por lo tanto, supongamos que tenemos el texto plano: S_i = [B , U , E , N , O], donde su correspondiente hexadecimal es: Hex(S_i) = [42 , 55 , 45 , 4e , 4f]. Al ser buscado cada caracter *i* de S_i dentro de C_i, se regresará su correspondiente en C'_i (o $xABC_i$, según corresponda). Entonces, se obtiene como resultado el cifrado parcial en formato *pseudo-hexadecimal*: $CParcial_i = ([0x0Q , 0x0X , 0x0^ , 0x0E , 0x0A] + RELLENO)$. El vector de *RELLENO*, consiste en valores aleatorios hexadecimales y/o *pseudo-hexadecimales*, que tendrán que agregarse, en caso de que el vector S_i sea menor que C_i (o viceversa). Finalmente, deben extraerse solamente el par de caracteres hexadecimales (o *pseudo-hexadecimales*), regresando una salida similar a la siguiente: $CifradoParcial_i = [0Q , 0X , 0^ , 0E , 0A] + RELLENO$ (Rangel et al., 2023).

La *tercera parte* o *fase* del procedimiento del denominado: *Noised random pseudo-hexadecimal GAs*, según se explica en la referencia (Rangel et al., 2023), corresponde al empaquetado final, en el cual, son almacenados de manera intercalada los tres vectores procesados previamente, en la primera parte y segunda fase. Por lo tanto, se incorpora al mensaje final, el contenido del alfabeto primario en formato hexadecimal, concatenado con los valores en *pseudo-hexadecimal* que contiene el alfabeto secundario, de cifrado de datos; y se concatena al final, cada caracter de manera intercalada, el cifrado parcial, también en formato *pseudo-hexadecimal*. Se explica en Rangel et al. (2023), que estos vectores incluyen un mínimo de 120 caracteres, por la extensión del alfabeto que utiliza *mod 120*, incluso si el texto plano de entrada es más corto, se rellena con "ruido" en codificación hexadecimales y/o formato *pseudo-hexadecimales*, según se prefiera, pero deben seleccionarse aleatoriamente (con reemplazo). Pero, si el texto plano de entrada tiene una extensión mayor a 120 caracteres, entonces se tendrán que rellenar de manera aleatoria (con reemplazo), los alfabetos: primario y secundario, usando valores hexadecimales y/o *pseudo-hexadecimales*, respectivamente. El empaquetado final, se define formalmente por: $MensajeFinal = C_i + C'_i + CifradoParcial_i$, donde el operador + , significa concatenación (Rangel et al., 2023).

Se puede observar que la primera fase del procedimiento con *Noised random pseudo-hexadecimal GAs,* es muy extensa o robusta, y por ende, puede demorar muchas generaciones para alcanzar la convergencia, o incluso, nunca lograr concluir esta etapa. Es por tal razón, que el alcance y objetivo de la presente investigación, es reducir tiempo del cifrado de datos y evitar que la ejecución quede detenida en un ciclo infinito. El nuevo método propuesto, ha sido denominado: *algoritmo aleatorio con ruido pseudo-hexadecimal*: *noised random pseudo-hexadecimal* (se ha eliminado el calificativo *GAs*), el cual, es recomendado como nueva alternativa en el cifrado dinámico, comparando resultados de ambas metodologías: la que utiliza GA y la nueva propuesta (sin uso del GA); esto es, bajo la hipótesis de que este nuevo método permite alcanzar la convergencia en menos tiempo sin hacer uso del algoritmo genético, y por ser un algoritmo de cifrado de datos dinámico, es capaz de producir distintos resultados, aunque utilice la misma entrada de texto plano. Lo anterior, sustentado en una reciente investigación relacionada, que fue presentada por Rangel & Rangel (2024), donde se propone un algoritmo para el cifrado de datos, basado en estrategias de *mutación*, con la finalidad de camuflajear el texto cifrado, que no utiliza el proceso completo de los GA, porque solo emplea la mutación por sustitución al 50% del texto cifrado (omitiendo el proceso de selección, cruce, y evaluación con función de aptitud; que comúnmente se aplica durante el procedimiento estándar de un GA). Este algoritmo, ha sido reportado por Rangel & Rangel (2024), que se comporta rápido en el cifrado de datos y utiliza procesos basados en métodos aleatorios, y por ende, se considera un algoritmo de cifrado dinámico y seguro para las organizaciones. Cabe recordar que el cifrado de datos dinámico, produce distintos resultados, aunque utilice la misma entrada de texto; y ello, puede confundir a los ciber-delincuentes. Finalmente, la justificación del presente trabajo, es debido a que, el formato *pseudo-hexadecimal*, se considera una propuesta muy prometedora, pero ha sido poco estudiado en la literatura. Incluso, existen dudas al respecto de dicho formato, porque Rangel et al. (2023) no presenta una formulación matemática o información completa al respecto de dicho formato. En esta investigación, se propone dar aportación empírica a favor del formato *pseudo-hexadecimal*, presentando una tabla de equivalencias: ASCII, hexadecimal, pseudo-hexadecimal. Por último, se introduce una modificación al *algoritmo genético aleatorio con ruido pseudo-hexadecimal,* con el propósito de mejorar su rendimiento. La comparación de resultados, también es presentada en términos de precisión del sistema, empleando la modalidad de validación cruzada para poder contrastar la información con otras investigaciones relacionadas (Rangel et al., 2023; Rangel & Rangel, 2024).

2. METODOLOGIA

La presente investigación, fue desarrolla en las instalaciones del Tecnológico Nacional de México, en el campus Instituto Tecnológico de Ciudad Altamirano, en el Municipio de Pungarabato, Estado de Guerrero, México. El tipo de investigación relacionada con este trabajo, se considera: cuantitativa, experimental y exploratoria, debido a que, el uso de la codificación pseudo-hexadecimal (Rangel et al., 2023; Rangel & Rangel, en revisión desde 2024), ha sido poco estudiado en la literatura.

2.1. DISEÑO DE LA INVESTIGACIÓN. NUEVA PROPUESTA: NOISED RANDOM PSEUDO-HEXADECIMAL

El método de cifrado de datos, diseñado como nueva propuesta en la presente investigación, es aquí denominado como: *noised random pseudo-hexadecimal* (*algoritmo aleatorio con ruido pseudo-hexadecimal*), eliminando el calificativo GAs. Se trata de una modificación al método: *Noised random pseudo-hexadecimal GAs*, con el propósito de omitir el uso del algoritmo genético, y de este modo, evitar caer en un ciclo infinito, haciendo más rápido el proceso de encriptado de datos, sin poner en riesgo la seguridad de la información en las organizaciones. Esta modificación, consiste en generar los alfabetos de cifrado/descrifrado, siendo seleccionados de manera aleatoria, sin seguir el proceso del GA, que consiste en la selección, cruce, mutación y evaluación con función de aptitud; todo ese procedimiento ha sido omitido en el procedimiento del nuevo método denominado: *noised random pseudo-hexadecimal*, propuesta que también utiliza módulo 120 (mod 120). Además, se introduce un nuevo alfabeto adicional, limitado con caracteres de prioridad, que son especificados por el cripto-analista de

manera anticipada, para evitar la selección de valores ASCII que no son imprimibles en pantalla, teniendo como resultado, textos cifrados más nítidos.

Cabe mencionar, que la primera definición de este tipo de propuesta, no se ha publicado en la literatura, de manera formal, solo ha sido presentada de manera local, en una conferencia o ponencia (Rangel et al., 2023), como propuesta de trabajos futuros; así como, en un reporte de investigación de manera local o institucional. Por lo tanto, todavía existen dudas al respecto de la implementación, así como, algunas ambigüedades. Un segundo trabajo realizado por Rangel & Rangel (en revisión desde 2024), aún no ha sido publicado, porque se encuentra en revisión, pero interpreta la implementación del denominado: *noised random pseudo-hexadecimal* como un algoritmo de sustitución, lo cual, se percibe que se trata de una modificación o nueva propuesta derivada de la versión original.

En la presente investigación, se implementa el método de cifrado en su versión estándar, de acuerdo con la información obtenida de un reporte de investigación local o institucional (Rangel, 2024b), el denominado: *algoritmo aleatorio con ruido pseudo-hexadecimal* (*noised random pseudo-hexadecimal*), que se le identifica como primer serie del siguiente modo: *noised random pseudo-hexadecimal I*, el cual, ha sido implementado usando un método heurístico. Se comenta que dicha propuesta, es una variante del algoritmo: *Noised random pseudo-hexadecimal GAs*, cuya diferencia es que se omite en el nombre el calificativo GAs, porque la nueva propuesta, sustituye el uso del algoritmo genético por un algoritmo heurístico. La razón de esta modificación, se debe a que: *Noised random pseudo-hexadecimal GAs*, en algunos experimentos, se observó, demoraba demasiado tiempo en generar los alfabetos de cifrado/descifrado. En ese mismo reporte de investigación (Rangel, 2024b), señala que el proceso del denominado: *noised random pseudo-hexadecimal*, es el mismo que su predecesor que hace uso de GA, ya que, ambos consisten en tres fases o etapas. La diferencia radica, en que la nueva propuesta: *noised random pseudo-hexadecimal*, realiza una variante en la primer parte o fase, mediante la simplificación del proceso de selección de los alfabetos de cifrado/descifrado, omitiendo el uso del algoritmo genético, y seleccionando ambos vectores de manera aleatoria sin reemplazo, siendo comparados entre sí, y finalizando el proceso de selección al comparar cada posición *i* de ambos vectores, resultando diferentes absolutamente todas las localidades de los denominados: C_i y C'_i, que corresponden a los alfabetos que permitirán el descifrado y cifrado de datos, respectivamente. Del mismo modo, se señala que esta condición, no es obligatoria, ya que, podría permitirse que algunos caracteres ASCII tuvieran el mismo valor, en la misma localidad, tanto en C_i como C'_i, pero que no tuvieran demasiada ocurrencia. Por ejemplo, de los 120 caracteres del alfabeto, podrían repetirse 5 o 10, pero no 50 o 100, porque ese detalle podría hacer que el mensaje cifrado fuera intuitivo para ser descifrado, aunque se intuye que esta situación no representa riesgo a la seguridad del mensaje cifrado, si se aplica un proceso posterior de desplazamiento. Cabe mencionar, que usando la nueva propuesta denominada: *noised random pseudo-hexadecimal I*, el proceso de selección de los alfabetos, no requiere de muchas repeticiones, debido a que, en el algoritmo heurístico se establecen reglas de búsqueda finitas, que siguen el orden de ideas que se describe enseguida: Primero, se define un vector denominado: _ABC que contiene la secuencia de caracteres válidos a usar. Se trata de un nuevo alfabeto adicional, que está limitado con caracteres de prioridad, los cuales, pueden ser alfanuméricos en mayúsculas y minúsculas, si se desea distinguirlas; así como, algunos caracteres de ruido, pero que deben ser imprimibles en pantalla y en archivo. En este contexto, sobre la inyección de ruido, se debe ser cuidadoso en la selección de caracteres que formarán parte de los alfabetos de cifrado/descifrado; sobre todo en el denominado: _ABC, si se incorpora elementos que estén fuera de la tabla ASCII. Por ejemplo: El uso del caracter '▒' que se supone su correspondiente ASCII es 178 (pero al ser convertido a entero u ordinal regresa el valor: 9619). Otro caracter, que se ha utilizado, con el propósito de inyectar "ruido" en los alfabetos de cifrado/descifrado, es el caso de: '□'. Dicho caracter, fue extraído del nombre de documentos que en sistemas Android no reconoce al migrar datos de Linux o Windows a la plataforma Android. Al parecer, ese caracter no tiene un correspondiente en la tabla ASCII. Se realizaron pruebas en lenguaje de programación, donde se obtuvo el valor entero u ordinal: 65533. El código de Python 3, utilizado para dicho propósito fue el siguiente: c1 = "▒" ; c2 = "□" ; print("->" + c1 + " = " + str(ord(c1))) ; print("->" + c2 + " = " + str(ord(c2))). Se hizo uso de dicho caracter en el vector: _ABC, con la finalidad de inyectar ruido, y no se reportaron problemas en la fase de

experimentación. Posteriormente, se procede a generar los vectores vacíos denominados: ABCD y xABCD con *mod 120*. La diferencia de C_i con la nueva propuesta ABCD, radica en el tipo de datos, ya que, este último vector puede contener valores hexadecimales y *pseudo-hexadecimales*, de manera simultánea. Para el caso de C'_i en comparación con el vector: xABCD, se presenta la misma situación. Por lo tanto, la selección del vector final: ABCD se realiza buscando (aleatoriamente) cada caracter *i-aleatorio* de _ABC en vector ABCD, que en un principio estará vacío. Si el caracter $\_ABC_{i\text{-}aleatorio}$ no se encuentra en el vector ABCD, se agregará en este mismo, pero usando un valor en formato: *pseudo-hexadecimal*. Si el vector _ABC es de longitud menor que 120 posiciones o localidades, en este caso, se tendrán que rellenar el resto de las localidades de ABCD con números aleatorios en formato hexadecimal, siempre y cuando, dicho caracter a agregar no se encuentre en el vector ABCD, de lo contrario se debe generar un nuevo número aleatorio hexadecimal, hasta sortear alguno que no exista su equivalente: *pseudo-hexadecimal*, que haya sido agregado previamente en ABCD. Al finalizar, se tendrá un vector con datos hexadecimales y *pseudo-hexadecimales*, sin redundancias, lo que sustituye a un proceso aleatorio sin reemplazo, similar al generado por el algoritmo genético, pero que demora menos tiempo en su generación. Para obtener el vector xABCD, se realiza el mismo proceso utilizado para generar ABCD. Si lo desea, puede comparar ABCD con xABCD, para determinar el grado de similitud y proceder a generar nuevamente los vectores, en caso de que coincidan exactamente. Aunque en la práctica, se ha observado que es muy difícil que esa situación ocurra, y además, aunque fueran exactamente las mismas secuencias ABCD y xABCD, generalmente llevarán valores aleatorios hexadecimales, lo que hace difícil a las herramientas de descifrado (e incluso a los ciber-delincuentes) la interpretación del mensaje encriptado; y si ha ello le agregamos, que en la tercera fase, se incorpora ruido en el mensaje con cifrado parcial, se tiene grandes posibilidades de ser diferentes ambos alfabetos o vectores, lo que produce un resultado de difícil interpretación para los ciber-criminales. Finalmente, una vez obtenidos los vectores: ABCD y xABCD, estos sustituyen a los alfabetos denominados previamente como: C_i y C'_i , respectivamente; y posteriormente se sigue el mismo proceso del algoritmo: *Noised random pseudo-hexadecimal GAs*, partiendo desde la *segunda fase o parte 2*; y, concluyendo con el empaquetado de la *fase o parte 3* (omitiendo las tareas de la conversión a formato: *pseudo-hexadecimal;* debido a que, los vectores de alfabetos, incluyen ahora, no solamente valores en hexadecimal, sino también, cadenas *pseudo-hexadecimales*). Por último, para el descifrado de datos, se utiliza la información del mensaje final, separando los vectores (intercalados) y buscando el mensaje cifrado parcial en el vector C'_i regresando su correspondiente en *Ci*, convirtiéndose el hexadecimal y *pseudo-hexadecimal* a código de caracter (char) del ASCII.

2.2. MATERIALES, EQUIPO O RECURSOS UTILIZADOS

Con respecto a la unidad experimental, que fue empleada para realizar la implementación y pruebas, consiste en una sala de investigación, de la cual, solamente se utilizaron dos computadoras, con capacidad de 8 GB de memoria, espacio disponible en disco de 200 GB y velocidad del procesador con 2 Ghz, ambas instalado el sistema operativo Windows 10 (Microsoft, 2024) y lenguaje de programación Python 3 (Python.org, 2024). Adicionalmente, se hicieron algunas pruebas en un equipo de cómputo móvil con sistema operativo Android 12 (Android 12, 2024), con memoria de 4 GB, espacio de almacenamiento de 64 GB y velocidad del procesamiento de 1.7 Ghz. Se instaló la aplicación Pydroid3 (Pydroid3, 2024) para ejecutar código fuente escrito en lenguaje Python 3. El propósito del uso de este último equipo móvil, fue para corroborar resultados.

Cabe aclarar, que los datos utilizados son del tipo entero, hexadecimal, cadenas de texto en formato ASCII (American National Standards Institute, 1963) y UTF-8 (Unicode Consortium, 2022); así como, el denominado: *pseudo-hexadecimal* (Rangel et al., 2023; Rangel & Rangel, en revisión desde 2024); que en algunos casos, fueron almacenados usando arreglos (vectores) para su procesamiento. La información consiste en secuencias de texto plano y texto cifrado. El texto plano, se trata de la secuencia de caracteres ASCII (American Standard Code for Information Interchange) o UTF-8 (Unicode Transformation Format 8 bits) que se le aplica el cifrado de datos. Dichas secuencias, en ocasiones fueron convertidas a entero (ordinal ASCII) para procesamiento del algoritmo o método de cifrado que más adelante se describe. Una vez cifrados los datos, se convierte a hexadecimal el alfabeto

base, y a *pseudo-hexadecimal* el alfabeto secundario, junto con el resultado cifrado. Es por tal razón, que el texto cifrado, tiene mezcladas secuencias en formato hexadecimal y *pseudo-hexadecimal*. El alfabeto secundario, guarda el equivalente cifrado y el alfabeto base (o primario) es el diccionario que permite lograr el descrifrado de datos (por sustitución y/o desplazamiento). Con respecto a la recolección de datos, fue realizada de manera aleatoria con reemplazo (porque incluye repeticiones de los ejemplares). Mientras que las selección de los alfabetos se hizo utilizando el método aleatorio sin reemplazo, sin repeticiones (ello se explica más adelante). Por lo tanto, se diseñaron varias secuencias de texto plano (aleatoriamente), siendo incorporados mil ejemplares en una muestra de entrenamiento (Rangel, 2002, 2022; Rangel et al., 2023), acompañadas de su respectivo texto cifrado. Cada ejemplar de texto plano (es un vector ASCII o UTF-8), que se comporta como etiqueta de clase, tiene un tamaño máximo de 255 posiciones, para poder comparar resultados con otras investigaciones (Rangel et al., 2023; Rangel & Rangel, 2024). La longitud del texto cifrado, puede ser mayor, dependiendo del tamaño del texto plano, así como, el método que se haya utilizado para cifrado de datos. El texto plano, que es la secuencia de entrada de datos (y etiqueta de clase), se procede a encriptar y almacenar en la muestra de entrenamiento, acompañado del tiempo de cifrado y descifrado. Estas secuencias de texto, usan formato ASCII cuando son cadenas simples, por ejemplo, suponiendo el uso de una contraseña sin caracteres especiales como el caso de: "Welcome". Sin embargo, para texto que incluye carácteres especiales como: "▒Welcome▒", los valores ordinales (enteros) de los caracteres: "▒" y "", no se encuentran en el rango de la tabla ASCII, debido a que, su máximo valor es 255; y para el primer caso, el número ordinal es: 9619. Teniendo para el segundo caso, un valor ordinal de: 65533; ambos valores exceden el máximo de 255 de la tabla ASCII (American National Standards Institute, 1963). Este tipo de situaciones fueron controladas usando formato UTF-8, siendo indicados en la codificación *pseudo-hexadecimal* como: 0▒ y 0, respectivamente; para poder "camuflajear" un poco, mezclando con otros números hexadecimales (del alfabeto base); así como, inyectando ruido mediante la incorporación de otros caracteres seleccionados aleatoriamente (con reemplazo), simulando el esquema *pseudo-hexadecimal*. Lo anterior, para confundir a los ciber-delincuentes, ya que, no pudo ocultarse este tipo de detalles con formato UTF-8.

2.3. PROCEDIMIENTO

En lo conciente al modo en que se ha realizado este estudio con el formato *pseudo-hexadecimal*, inyectando ruido en tareas de cifrado de datos, fueron llevadas a cabo las actividades, empleando el siguiente orden de ideas: Primero, se prepararon los datos, utilizando métodos aleatorios con reemplazo. Posteriormente, se aplica cada algoritmo o método de cifrado/descrifrado, utilizando una variante del método validación cruzada, con cinco repeticiones, tal como se ha empleado en estudios previos relacionados con la encriptación de datos, todo con la finalidad de poder comparar resultados con otras investigaciones (Rangel et al., 2023; Rangel & Rangel, 2024). Las variables utilizadas en la fase de experimentación fueron: el uso de cadenas o vectores de texto cifrado; así como, el error ocurrido y los tiempos de cifrado/descrifrado de datos. Considerando que, si un texto encriptado, al ser descrifrado por un método, no coincide la secuencia de desencriptado con el texto plano de entrada, es anotado como error. Finalmente, se calcula el promedio (y su respectiva desviación estándar) del error ocurrido y los tiempos de respuesta de cada método de cifrado de datos, de manera separada, teniendo en cuenta los cinco experimentos o repeticiones realizadas. Cabe recordar, que cada uno de los métodos de cifrado utilizados, fueron evaluados con la misma muestra de entrenamiento, separadamente. Tal como se ha mencionado anticipadamente, se utilizó una modificación de la validación cruzada (Rangel & Rangel, 2024) como método de estimación del error, con cinco repeticiones, usando un 20% como muestra de control y el 80% como muestra de entrenamiento, para evaluar el modelo. Al concluir el proceso de validación cruzada, se ha calculado el promedio y desviación estándar de los tiempos de ejecución de cifrado/descrifrado, así como, el porcentaje de error.

Con respecto a la implementación del *Noised random pseudo-hexadecimal GAs*, se realiza una modificación al proceso en la segunda fase, porque se omite el uso del vector de desplazamiento K_i, llevando a cabo solamente un procedimiento basado exclusivamente en sustitución, tal como se describe enseguida: (i) Calcule *MATRIZ* =

getNoisedCypherGAsDictionary(). (ii) Asignar *ABCD* = *MATRIZ*[0]. (iii) Asignar *xABCD* = *MATRIZ*[1]. (iv) Sea S_i el texto plano de entrada. (v) Calcule C_i = *getSustituto*(*ABCD, xABCD,* S_i). (vi) Calcule *Paquete* = *getPaquete* (*hexa* (*ABCD*) , *pseudoHexa* (*xABCD*) , *pseudoHexa* (C_i)). DONDE: El vector *ABCD* guarda los valores del alfabeto primario, mientras que *xABCD* contiene los valores del alfabeto secundario. El vector S_i es el texto plano. La función *getSustituto*, regresa el cifrado parcial (C_i) de la entrada S_i, considerando que si una ocurrencia de S_i se encuentra en $ABCD_i$, entonces se asigna C_i=$xABCD_i$; de lo contrario se conserva el mismo caracter, pero en *pseudo-hexadecimal*: C_i = *'0'*+S_i. La función *getPaquete* permite concatenar, en forma intercalada, los tres vectores: *ABCD*, *xABCD* y C_i. Además, resuelve cuántos elementos o posiciones debe incluir como *RELLENO*. La función *hexa*, garantiza que el vector *ABCD* se guarde en formato hexadecimal, mientras que la función *pseudoHexa*, asegura que los vectores: *xABCD* y C_i contengan valores *pseudo-hexadecimales*, con inyección de ruido. Finalmente, la función *getNoisedCypherGAsDictionary*(), ejecuta el algoritmo genético para cifrado de datos con ruido, retornando dos vectores, los cuales, representan al alfabeto primario y secundario. La forma de implementación fue la siguiente: (1) Sea *ABC* y *xABC* dos vectores con números enteros, seleccionados aleatoriamente con reemplazo, cuyo contenido representa un valor ordinal de la tabla ASCII con *mod 120*, es decir, iniciando en el ASCII=30 y terminando en el valor 150. (2) Poner *n* = 100. (3) Sea *ME* un arreglo bidimensional, con 100 filas y 121 columnas, seleccionados sus valores ordinales, de manera aleatoria con reemplazo. Cada fila representa un posible alfabeto, la última columna representa la etiqueta de clase, la cual, se asignará un valor inicial de cero. (4) Poner *Generación = 0* ; y asignar *P = -1* ; (5) Mientras *P* sea diferente de cero, hacer: (6) Mientras (*Generación* <= 100) hacer: (7) Ejecutar *GAsSeleccion* (*Generacion, ME, ABC, xABC*), procedimiento que permite etiquetar las filas de la ME con clase 1 y clase 2, esto es, si la distancia de *ABC* es menor o mayor, respectivamente, considerando cada par de filas en *ME*, elegidas secuencialmente. Al final, se elimina de la *ME*, aleatoriamente, el 25% de filas por clase. (8) Se ejecuta *GAsCruce*(*Generacion*, *ME*), procedimiento que permite cruzar el 50% de filas en la *ME*, usando uno o dos puntos, aleatoriamente. Recuperando el 50% de las filas eliminadas previamente, siendo etiquetados como clave=0, estos nuevos elementos.(9) Se ejecuta *GAsMutacion*(*Generacion, ME, ABC, xABC*), procedimiento que permite la mutación del 10% o 20% de las filas, mutando aleatoriamente, por intercambio o por selección de un valor, excepto la etiqueta de clase. También, se aplica mutación a los vectores *ABC* y *xABC*. (10) Se avanza a la siguiente generación, es decir, se asigna: *Generación=Generación+1* ; (11) Si la *Generación* es mayor o igual que 100; se asigna *Generación=0*; y se rompe el ciclo mientras del paso "6"; (12) Se evalua la función de aptitud: *P = getAptitud*(*ABC, xABC*) y regresa al paso "5". La función *getAptitud*, devuelve la suma de las coincidencias de *ABC* localizadas en *xABC*. (13) Si demora demasiado tiempo, ya no se evalúa en paso "5" la sentencia *P diferente de cero*. En su lugar, se evalúa: Si *P<=5* o *P<=10*. (14) El algoritmo termina retornando los vectores: *ABC* y *xABC*, que corresponden a los alfabetos primario y secundario, respectivamente. Este procedimiento se aplicó a cada patrón de la muestra de textos cifrados, haciendo uso de la validación cruzada modificada (Rangel & Rangel, 2024). Un problema que se observó con este algoritmo, es que hace vulnerable la seguridad de los datos; ya que, puede caer en un ciclo infinito, y por ende, producir errores al momento del descifrado de datos. Para el caso de la nueva propuesta: noised random pseudo-hexadecimal, se aplicó también, de manera separada, el mismo proceso de validación cruzada, con la misma muestra de textos cifrados.

2.4. ANÁLISIS DE LOS RESULTADOS

En la fase de experimentación, fueron realizados varios experimentos utilizando los métodos de cifrado de datos: *Noised random pseudo-hexadecimal GAs* y *noised random pseudo-hexadecimal*, siendo empleados separadamente, tal como se ha descrito previamente, en la sección de metodología. Cada experimento fue llevado a cabo usando la misma muestra de entrenamiento (ME), que ha sido generada de manera aleatoria con reemplazo, donde cada fila es un ejemplar que guarda texto cifrado, con su respectiva etiqueta de clase, que corresponde al mismo texto plano de entrada. El número máximo de filas en la ME, tal como se ha comentado anticipadamente, fueron mil ejemplares almacenados, con tamaño máximo de 255 caracteres para el texto plano. La ME incluye otras dos columnas adicionales:

una para guardar el tiempo de cifrado y otra columna para conocer el tiempo de descifrado, ambos medidos en milisegundos. Si el texto cifrado, al ser desencriptado, no coincide con la entrada de texto plano, es anotado como error. Para calcular el error y estimar la velocidad promedio del cifrado/descrifrado de datos, se ha utilizado una variante de validación cruzada con cinco repeticiones, la cual, fue propuesta en otras investigaciones (Rangel et al., 2023; Rangel & Rangel, 2024), todo con el propósito de poder comparar resultados. Primero, se realizaron los experimentos con el algoritmo genético denominado: *Noised random pseudo-hexadecimal GAs*, utilizando alfabetos con módulo 120 (mod 120), utilizando la ME. Se obtuvo el *promedio* (*media aritmética*) y *desviación estándar* de los tiempos de cifrado y descrifrado con esta alternativa, donde se observó en varios experimentos que hubo errores en el descrifrado de datos, y además, el procesamiento para cifrado de datos, en ocasiones entraba en un ciclo infinito.

Tabla 2: Resultados preliminares de los métodos de cifrado dinámico aleatorio con ruido *pseudo-hexadecimal* (un ejemplo usando como texto plano: ▒Welcome▒)

Método de cifrado de datos	Operación	Experimento #1	Experimento #2	Experimento #3	Experimento #4	Experimento #5	Promedio (desviación estándar)
Noised random pseudo-hexadecimal GAs	Tiempo de cifrado (milisegundos)	3890.7663	2153.5804	2119.5969	*2061.8393*	2113.9993	2467.95644 (796.05140)
	Tiempo de descrifrado (milisegundos)	12.8488	12.2947	*11.8539*	11.9721	12.1018	12.21426 (0.3905955)
	Porcentaje de error	0	1	0	0	0	0.2 (0.4472135)
Noised random pseudo-hexadecimal I (versión estándar)	Tiempo de cifrado (milisegundos)	114.70955	108.76200	100.52121	**100.49051**	107.46307	**106.389268** (6.0257665)
	Tiempo de descrifrado (milisegundos)	10.819097	11.047939	10.610293	11.187608	**9.5422308**	**10.64143356** (0.6527507)
	Porcentaje de error	0	0	0	0	0	**0** (0.0000)

En la Tabla 2, se presenta una muestra de un experimento, mientras se procesaba la cadena de texto plano: ▒Welcome▒, en el experimento #2, se observó no solo que ocurrió error en el descrifrado, sino también, se tuvo que forzar la salida del sistema por congelamiento, lo que determina haber entrado el algoritmo genético en un ciclo infinito. Este tipo de situaciones fueron muy recurrentes, durante la etapa de experimentación para este método basado en GA (en la Tabla 2, se observa un porcentaje del 0.2 de errores ocurridos en promedio). Posteriormente, se aplicó el nuevo método, aquí propuesto, denominado: *noised random pseudo-hexadecimal,* también usando *mod 120* y empleando con cinco repeticiones la misma validación cruzada, antes referida.

Tabla 3: Algunos ejemplares de muestra seleccionados de los textos cifrados por los métodos de encriptado dinámico aleatorio con ruido *pseudo-hexadecimal* (cada secuencia corresponde a un experimento, cuando ha sido utilizado como texto plano: ▒Welcome▒).

Método de cifrado de datos	Número de muestra	Texto cifrado (muestra seleccionada)
Noised random pseudo-	#1	270 0□450C0>460u0p4c0~0E6200^390*0/940.0Q420 0p2b0Z0□8d0#07706fd280 fe5c0 ff820 fe560Aff340Nfd600·fd 8e0hfd670)fe430Kfe330wff8c0 ff540 ff6c0Efe7e0Sfd6a09fe260_ff610efe210 fe3f0Jfd2a0Tfe750|fd480}fec50□fe410[fe 5b0"fe4a0'ff890

hexadecimal GAs		fd4d0`fd3d0Bff920ofd780,fe630^fd6b0 ff2d0afd950<fe590Iff440bff570>ff350⇔ff490kfd8b0Mfd360Yfd700nfe380□ff310Rfd2c0Lfd9103fd470sfd4f0ffe3c0+fe7d0xfd3b0$fe6d0Qfe790dff240Ufe810□fe500mff7f0Hfd7c0□fd6408fd230%ff8605fe720 Ffd4b0Pfe830@ff7b04ff1f0vfd5f0 ff200Gfe580{fe370 fd6e0□ff2e0rfd800Xfdba0&fe4e0□fd660-fd220gfe290-fe8a0jfd3a0lfd5e0□fe930qfe900!fd8f0(fd730\ff690\ff3e0□fe5a0□fe650pfe6f0/fe710;fe880yff760?fd550ife320)fe5d0Dff1e0Wfe520=fd2500fe400 fd7a0Vfe850Off870tfd740 ff2f0]fd300 fd680cfe510zff
	#2	2f0 0□370R0.5a0]065d0 0l210*0 850I0b3a0⌋0 90000642080□7d0 07604fd470Jfd310yfe7c0_ff5203ff6402fd660 fd920 fd6e0,fd270 fd560tfd410~fd570.fd810"fe8f0%ff940Bfe630 fe440vfe7b0lfe6b0&ff480 fd680afe590cfe350 fd750)fd780+fe320Wfd530ife8a0pff870!ff580□fe390□ff4b0^fe340Ffe260#fe5c0Ufd540⌈ff5f0Zffa90dfd3f0sfe0d0/fe4a0@ff740Qfe950\ff8b0 ff220kfe250rfdb80;fd670ufe4e0Gff880□fd450-fd840 fd720⇐fd610-fd8c0Afe730ofd860fd6d0□fd3c0Kfe3b0∅fd930□fe490<fe4f0Xff200 fd5b0↓fe360fd290zff5501ff2d0□ff6506ff7f0Pff460⇔fd400$ff4d0:fd910gfd1f0□fe5e0□ff7a0?ff820Ofe300(fd890Nfe700≥fd2c0□fe8005fd6f0bfe2a0wff240'fd690qff3d0⌈ff430nfe3307ff6a0=ff8d0□fe8e0Mfd3e0}fe600Vff280⌊ fd4c0[fd7e0Lff1e0jfd230`fd790eff830□fd770Hfd510|ff620Cff710Efd500Yfe2e0>fd
	#3	6d0-0□940X0↔790 0]530_0l590 0G20050 400⌊0-370@0]90030□7e0 0280'fd8d0:ff8f0$fd5a0 fe350 fdcd0fe5c0 ffea0.ffb609fe520\fd310 fd8e0 fd480Yff4707fe160%fe360Eff690 ff260Sfe560qfe430Ufd830Pfd1f0 ff750|fd670Jfd730 ff5d0"fe1c0tff3d0>fe8b0wfd3a0□fd850□fe800Tfe410,fd2f0Nff290afd4c0;fe910?fe500!fe390 feb20Ffe7a0□fe4a0[fe630Gfd740mfd3f0□ff7c0jff600offb70Dfd610iff2d0&fd5b0dfe880□ff3b00fe5f0#fd810~fd660Lfdd00pfd2101fd230gff340□fd510Bffc00Vfd6a0+ff890Mfe640*fe3e0rfe4b0sfe380Wff6b0^fd8c04fe460yfd7208fd7b0/ff220ff570↔ff7d0Cff700Qfd680eff2c0□fd870 fd5e0Hff840zfd7f0xfe710}fe250=fe550⊕fd920ufd760(ff2b0□fe4e0fd990\fe770<ff450Aff950`fd240Ofe1e0kff4f0bff620{ff540□fe2e0'fd4d0Zfd650]fe320)fd6f0 fd4206fd300Kfe930ffd860hfd090□fe820Rff
	#4	4b0C0□4e0P0-250@0w830-0n4f0N0;5a0i0T7a0p0)260Q0w7f050□7e0"0a50Lfe550 fd7c0zfd740cff620Vff5b0 ff310#ff700*fe100 fd920.fd1f0♥fe680Gff240 ff710gfe410 ff460/fe7d0_fd4d0 ff440?ff210Xff670mfe860vff750 ff3c0 fd8d0%fe2c01ff6b0 fe4802ff820off720Sfd8b0&fe430hff840)fd640\ff590□fe220ff290{ff8c0□fe7308ff8e0qfd660Wfd4c07fe6a0efe3d0□ff390yfe880 fe950>fe8a0lff5003ff330[fe320Dff520Mfd540`fd450Rfd9b0bff3404fe300ffe350Ffe190□fe360□fd780$fe910,fe530afe580^fe870xfe1e0<fe2d0]fd280|fd270Zff4909fe850Ife3706ff5d0Bfd2b0Efd6c0nfe610dfe420□fe760Kfe3e0:fe2f0□fda90Hfe5f0rfe6d0)fd790 fe810~fd1a0 ff940+fd3a0Jfd570-fe5c0□fe510ufe900sfd770!ff8f00ff800Afd650wfe4a0Yfd630;fd930 fe230jff7b0tff2a0Ufd600=ff3b0Ofe400 fe6f0Tfd690'fd3f0 ff890kff560(fd
	#5	eb0#0□360a0R5600n870W0q770`0w380s0 470&0'41040n650n0□950o0500!fe8f0Bfe3c0cff820"ff2d0 fe630wff900mfd6e0efe4c0vfe2307fe860hfe4f0 fe260Sfd2e0^fe7f0 fe2901fef50 fd8e0 fe6a0Kfd8a0⊂fe320 fe940Lfe6b0 fe8c0=fe580Off600 fe710$fd680)ff5e0rfe930 fd5d0/fd840ff4d0[fe670xfd8b0□fe780@fd7b0□fd4e0□ff700uff7a0Qfe720Efd760,fd1f0Cfd2c0{ff4a06feb00Hfe350Tfd280yfd4b0Xfe850]fd8902fd610□fd620Fff7c0Yfe6c0qff250pfd3b0bfd690fff400Pfe520Dfe3d08fe430+fd220lff1e0kfd240|fd5b0÷fe660□ff3f0Afe5a0Vff490%fe3903ff530Gff550jfe570Rfe270fe910dfd8100fd210 fd7e0}fe370⇓fd2a0Zff2f0Jfd540?fd800(fe740tff920<ff300 fd200Nff510□fd480ff310□fe8d0gfd6d0'fd5c0-fe750ife2b09fd6f0 fe880-fe5f0~fd340.fd3a0Ife440Mfe330 ff3e0\fe460;fd640 ff7d0*fd450□fe730:ff830□fd
Noised random pseudo-hexadecimal I (versión estándar)	#1	75b10▓030f480m54b606327b0i0C90200e0Y280J540Y0Qb6640R0▓0ñ085f3c2606e0Iff0Ñ0¥ff0o0Yff38f6fff20Dff070Xff0L93ff0Z0Gff770zffc278ff9b0tff0ne3ff0H38fd2c2ff0j0ff950qff0t02ff460dffa405ff0h69ff010affe003ff0v89ff44bcff0X0uff0Q0pffaa0Sff0u0bff0G0hff0D09ff0fbeff879ff0T27ff0¥d6ffdd42ff050Tff084aff305eff0b0kff990Vffbf0mff020sff6722ff0N0Kff4c2fff0w0Eff0R96ff0Fcaff5001ff0l7bffed0Uff000ñff0a8aff0g9dffb0f0ffbdfff740jff0eb6ff0B61ff0W48ff0E46ffa90Mff0yc1ffcb0wff5e0Lff630Pff7a0off0P0Fff0K0rff040cff260Nff0Ja3ff9c4fff6f0Offaf0lffc475ffb4bfff0pd5ff900vff090gff0k0HffBffac8dff0da2ff0Ve0ff8c7eff2d9afff83cffa56cffc70Aff7e0iff0I0Wff0M70ffb92eff0A2cff0S0Zff0O00ff0z07ff0s83ff0C0nff05fff0c90ff0q68ff3e0yff3706ff0r9effba0Ñff0U04ff60e9ff
	#2	524a0▓a5ec3a0D3100Ñ0Hdfcf0Ff5e0030t0o0t0v0Z0500Y020▓050mad0d0S008c0ff0C0gff800eff0y0Kff3bc6ff0ñ2dff36b7ff0701ff0G99ffcc7aff2f0dffc10Pff0R0nffbe0ñff0zabff0g42ff310Tff3a0cffda04ffc573ff665eff0s0Vff0Jcaff300jff0P07ff040Jff34d2ff0j0Lff0U41ff0adff0Odcff0t44ff683cff9608ff0X0Bff580Rffe478ff57fcffe909ff600Dff5ebdff0122ff97a4ffa457ff0Vacff6b5aff0n0Ñff0E66ff550Gff400uff0N0pff280Zffd10Iff0m0vff0H0aff0¥27ff0L0Offd22eff99afff0M0hff0391ffe59cff0F0iff0K0qff4200ff0qe9ff59d8ff0p0Yff06e3ff0f0Nff0a0zff0i0¥ff0r0Cffa80Eff0B0lff0Qe7ff0kd3ff530off0ldfff0S0sff090wff0T0Xff0W3affba0yff02eeff63ceff0e0ff9c0Mff0v0bff0u70ffdca6ff4d2ffff80kff0cf5ffc6edff0b98ff0026ffbbe8ffe20rff6106ffd80Qff0A0Uff210fff4a93ffea62ff0h0Aff0w0Wff0I5fffcee6ff
	#3	0b0a0▓c90c0Bf47ad40za80E0j0C0d8c0re5230n0O0c0dd40B990▓560Ube2506030bfff0Tcaff0d26ff0F0qffd40wffab07ff0w40ff0W0Bff3cd8fff50lffce22ff0beff86cbffa149ffeb0mffb20Mffa8ddff0n0yffa587ff050Xff9f09ff0J38ff9e0Affb80jff7d25ff0g0Sff93edff0m0Offa30bff0s0Kff0M7dff4996ff0f3fff0L0Pff8e0pff0C0gff92e4ff0K0zff07feffe25fffd50Dff2151ff5cdeff0Eebffbf0iff0330ff0A0Qffb959ff6686ff0¥a7ff090ñff0S0hff29e1ff0601ff0Ubcff5441ff58cdff0oe5ff0Qdfff6870ff0R0eff3a00ff53c1ffc60Yff2b7bff0p0Vffcf02ff0I5eff0y0Gff0h20ff0r0Rff0G0off0N0Fff010Jff0Z2fff0H0Lff0k28ff0ÑÑffaf0uff0u0¥ff0084ff0X44ff0t50ff0i0fff0q0ff0a0Iffe30kff0ñ79ff0O32ff3d0Hff04f6ff0V6dff4045ff3108ffd604ff0l0Eff0D0Nff0ed4ff0Y0vff64e7ff0P0sff025bffdf0tffa0d1ff0805ff710Zff0v0Tff1e0Wffe803ff
	#4	060E0▓0Ñ3e5205a90q605308000ie20da10r03fb0Lb50A0qa4ae0▓0a0s910Vad00Z0yff0B6eff0E0zff4790ffb031ff6597ff0J0Zff0Rf0ff0l08ff630dff0e0qff0Fa6ff6f28ff0teffff0b06ff0r0vff0Qecff0G0Vfffb77ffde0Sff9904ff0Ua4ff0N0¥ff330Yff091ff0v0Cffa565ff0160ff0Mf4ffadacff3909ff4f0Iffc0Ñfffacaff0¥0bff7f0Kffdbdcff920Rff0X87fffe0Wff0w05ff0o0rffef0Off0W52ff0n8eff0je5ff5303ffda0ff0u0tff04a3ffa20Xff0D07ff0C0hff0T0Bff0902ff02d8ff670nff7e9dff8c00ff0ne7ff210Qff0783ffe13dffb80aff640lff080cff0z0wff0Iccffc05dffc8aaff0m0Lff9b0Fff800Dff0h0Jff0O21fff00Nffb60pff0y8fffbc9fffd20kff0Kc8ff3d69ff3c0Tff0k0fffcadeff0q42ffa90eff0S0off0A8bff0i6fff0s01ff0P0mff0p0Mffeac1ff403bffa30jffab0uff0ce2ff0H7cff0f0ñff500gff300Gff7427ff0g7ffff322ff380Hff0Yd3ff360Uff0L0Pff
	#5	b10j0▓0V4d0u0O79f289a3514b050P0I410m0o0ma40s0Lf20w0Q0▓0p5b951e000Gf9ff0P0Dff0ve8ff0ba2ff0Y9aff0u75ff0U0iff0E0ñffbdc1ffe159ff0D0Eff0C31ff040eff03ebff0Z0Xffa60Mfffc0cff0t0tff7cafff3b2cffb307ffca0Hff070rff9881ffb20bfffb0¥ff0h0dff08d5ff0l51ff3991ff0z4ffff30pff4f0afff901ff4153ff0a8bff8d0wff0q0Affc03bff0Sc9ff6200ff0L0Uff0Q99ffd77cff0534ff090lff010Gff0009ff6e0vff8e28ffae20ff0X83ff5ed6ff370kff0¥61ffc90hffce0nffc40sff0r04ffd30Iff5d03fff18dff0c0Pffde45ff0k0zff0y63ff97a1ff0Ñ0Jff7a0Kff9b0gffdb52ffb90Tff9c0Nff230Bff0i66ff0A9dff0ma4ff0F43ff6b0Off0nbbff0R0yff0K0qffd208ff095ff0ñ02ffd10Ñff7da5ff060Rff0j0Fff0ef2ff0Jfef0N8eff0H7eff4df0ff0M0Vfff42eff0g92ff0B48ff0d06ffac0Zff0W0uffeb0Wff800Yff0fb9ff460offc60fff590Cff026fff0T0Sff

Al finalizar, también se obtuvo el promedio y desviación estándar de los tiempos de cifrado y descrifrado con esta nueva propuesta, donde se observó que un ningún experimento hubo error. Además, los tiempos para el procesamiento del cifrado y descrifrado de datos son mucho más veloces con *noised random pseudo-hexadecimal*, en comparación con la alternativa que usa el GA (ver Tabla 2). Los experimentos descritos previamente, fueron

realizados con un programa de cómputo implementado en lenguaje Python 3; así como, PyDroid3. No se observó diferencia significativa en los resultados y tiempos de procesamiento.

Aunque la mayoría de los experimentos tuvieron éxito, se considera que faltó profundizar más en la experimentación, ya que, solamente se utilizaron cadenas de texto plano con longitud menor o igual a 255 caracteres. Es importante mencionar que, a la nueva propuesta, se ha eliminado el calificativo GAs, debido a que, ya no se utiliza el algoritmo genético. Esta modificación, ya se ha comentado en la sección de metodología, donde describe que en lugar de utilizar el GA, los alfabetos son seleccionados de manera aleatoria, sin seguir el proceso del GA, que consiste en realizar la selección, cruce, mutación y evaluación con función de aptitud; todo ese procedimiento ha sido omitido en el procedimiento del nuevo método denominado: noised random pseudo-hexadecimal. De acuerdo con lo anterior, se consideran los objetivos cumplidos satisfactoriamente, porque se logró reducir el tiempo empleado para el cifrado y descrifrado de datos. Además, con la modificación del procedimiento, omitiendo el uso del GA, se evita que la ejecución del proceso de encriptado quede detenida en un ciclo infinito. Estos resultados pueden ser corroborados en la Tabla 2. Del mismo modo, se logró cumplir la hipótesis de la investigación, debido a que, la nueva propuesta, denominada: *noised random pseudo-hexadecimal,* permite alcanzar la convergencia en menos tiempo, sin hacer uso del algoritmo genético, y por ser un algoritmo de cifrado de datos dinámico, es capaz de producir distintos resultados, aunque utilice la misma entrada de texto plano. Está situación puede ser corroborada en los resultados presentados en la Tabla 2 y Tabla 3. En la misma Tabla 2, podemos apreciar que: *noised random pseudo-hexadecimal*, es más rápido en el cifrado de datos que la propuesta basada en GAs, con una diferencia de 2361.57 milisegundos en promedio. Aunque en el descrifrado de datos, no se reporta diferencia significativa, solo de un 1.57 milisegundos de diferencia (en promedio). La velocidad más rápida en el procedimiento de cifrado de datos con: *Noised random pseudo-hexadecimal GAs*, se observó mientras se procesaba la secuencia de texto plano: ▒Welcome▒◆, durante el cuarto experimento y fue de 2061.8393 milisegundos. En el descrifrado de datos con la correspondiente secuencia de texto cifrado, se observó mayor convergencia durante el tercer experimento con un valor de 11.8539 milisegundos. En cambio, para la nueva propuesta: *noised random pseudo-hexadecimal,* se obtuvo la velocidad más rápida de procesamiento en el cifrado de datos, durante el cuarto experimento y fue de 100.49051 milisegundos. Finalmente, la velocidad más rápida en el descrifrado de datos fue alcanzada durante el quinto experimento, reportando un valor de: 9.5422308 milisegundos.

Por último, los resultados encriptados con la nueva propuesta, aquí presentada, genera un texto cifrado más nítido. Es decir, para secuencias de texto plano que incluyen caracteres que al ser cifrados exceden los valores de la tabla ASCII, como el caso del ejemplar: ▒Welcome▒◆, se puede observar en Tabla 3, que la alternativa que usa algoritmo genético, al ser copiado el texto cifrado dentro de una tabla en formato de Microsoft Word, independiente si la fuente original cifrada fue extraída desde un archivo de texto o desde la misma interfaz del sistema que hizo el cifrado, podemos observar la existencia de caracteres no imprimibles en pantalla. Por lo tanto, si copiamos el texto desde la tabla en formato Microsoft Word, y posteriormente, lo intentamos cifrar en el programa de cómputo escrito en Python 3, tendremos como resultado que en la mayoría de los casos, el texto será descrifrado de manera equivocada. Ello no ocurre con la nueva propuesta, que omite el algoritmo genético. Lo anterior, se debe a que el denominado: *noised random pseudo-hexadecimal*, genera sus resultados encriptados, utilizando en su mayoría, caracteres ASCII que son imprimibles en pantalla.

3. DISCUSION

Partiendo de la base, de que se ha observado en dominios prácticos, que el robo digital de datos, puede ocasionar grandes pérdidas en las finanzas de las organizaciones. Siguiendo la común práctica, una alternativa para amortiguar este problema, es el uso de métodos para encriptado de datos. El problema de este tipo de estrategias, es cuando se utilizan algoritmos estáticos, porque para una misma secuencia de entrada de texto plano, corresponde una misma salida como resultado cifrado. Esta situación, según Rangel et al. (2023) puede poner en

riesgo la seguridad de los datos en las organizaciones, ya que, mediante el uso de IA con procesamiento de lenguaje natural es posible crear un diccionario que corresponda, por ejemplo, a cada una de las secuencias cifradas de una contraseña de 16 caracteres. Por lo tanto, se propone el uso de métodos de cifrado de datos dinámicos, porque producen distintos resultados de texto cifrado, aunque se utilice una misma entrada de texto plano. Ello es bueno, ya que, puede confundir a los ciber-delincuentes. En este mismo contexto, una propuesta reciente, recomienda el uso de algoritmos genéticos (conocida como: *Noised random pseudo-hexadecimal GAs*). La presente investigación, es continuidad de dicho trabajo, porque se observó que ese algoritmo genético, demora demasiado tiempo y su proceso de encriptado puede entrar en un ciclo infinito. Dicha situación no se considera segura para las organizaciones. Ello ha dado lugar a la pregunta de investigación que sienta sus bases en la hipótesis, de que es posible mejorar el denominado: *Noised random pseudo-hexadecimal GAs*, para permitir alcanzar la convergencia en menos tiempo sin hacer uso del algoritmo genético. Al respecto, en la presente investigación, se logró cumplir dichos propósitos. Se observa en la Tabla 2, la existencia de una nueva propuesta denominada: *noised random pseudo-hexadecimal*, la cual, evita el uso del algoritmo genético. Además, se logra hacer 23.19 veces más rápido el proceso de cifrado de datos y se evita entrar en el ciclo infinito, mostrando 0% de porcentaje de error, al ser cifrado/descifrado un texto con esta nueva propuesta. Lo anterior, permite dar aportación empírica a favor de los métodos de cifrado de datos, que hacen uso del formato *pseudo-hexadecimal*. Del mismo modo, debido a que, ha sido empleada una nueva modificación del método de estimación de error: *validación cruzada aplicado en el cifrado de datos*, ello permite analizar los resultados de forma crítica y poder compararlos con los presentados en otras investigaciones (Rangel et al., 2023; Rangel & Rangel, 2024; Rangel & Rangel, en revisión desde 2024).

Con respecto a las ventajas y desventajas de la metodología empleada, ya se ha comentado anticipadamente, que la propuesta que usa el algoritmo genético tiene la ventaja de ser un método para cifrado de datos dinámico, lo cual, es bueno para la seguridad de datos de las organizaciones; y además, permite inyectar ruido a los paquetes cifrados usando formato *pseudo-hexadecimal,* teniendo a favor que ello puede confundir a los ciber-criminales. Sin embargo, tiene la desventaja de entrar en un ciclo infinito al momento de procesar el cifrado de datos, lo cual, puede ocasionar errores al momento del descifrado de datos. En la Tabla 2, se reporta un porcentaje promedio de error del 0.2%, al respecto. En cambio, la nueva propuesta denominada: *noised random pseudo-hexadecimal*, tiene las mismas ventajas que su predecesor (de calificativo: GAs), agregando que, por omitir el uso del algoritmo genético se comporta más rápido en el proceso de encriptado y en Tabla 2, se reporta 0% de error. Otra desventaja que presentan ambos métodos, consiste en que pueden llegar a seleccionar caracteres fuera del rango de la tabla ASCII, los cuales, no son imprimibles en pantalla. Esta situación, puede ocasionar pérdidas de la información si se desea copiar un texto cifrado desde archivo hacia otros documentos, como el caso de un archivo en formatos del Microsoft Office. Cabe aclarar, que este tipo de casos no se observó con la nueva propuesta, ya que, al utilizar un módulo 120 y estar limitado por un alfabeto de caracteres de prioridad, los resultados cifrados fueron obtenidos muy nítidos. Una última desventaja que se puede mencionar, es que ambos métodos, aquí empleados, no permiten ocultar al 100% caracteres en formato UTF-8, lo cual, no es bueno, porque puede dar una idea, aunque muy vaga, a los ciber-delincuentes, que podría llegar a orientarlos durante el proceso de descrifrado de datos. Aunque cabe aclarar, que como los alfabetos del texto cifrado están en formatos: *hexadecimal* y *pseudo-hexadecimal*, por ende, los caracteres del texto original están camuflajeados. En este contexto, según Rangel & Rangel (2024), observan con su propuesta basada en "mutación", que aunque no se utiliza el *pseudo-hexadecimal*, se considera que el camuflaje es bueno para el cifrado de datos, al momento de referir que: "la reducción de la ejecución de los tiempos para cifrado/descrifrado de datos puede lograr que los ciber-criminales no demoren mucho en el proceso de desencriptado, pero si fuera este el caso, ellos deberían conocer prácticamente cada valor del desplazamiento aleatorio K_i , los cuales, también han sido camuflajeados previamente. Por lo tanto, esas tareas de descubrimiento de dichos valores podrían ser muy difíciles...". En otras palabras, el ciber-delincuente, prácticamente tendría que "adivinar" los valores de cada uno de los desplazamientos aleatorios o posición de cada caracter en los alfabetos que utilizan los métodos aquí evaluados. Del mismo modo, el uso del diccionario creado con IA mediante procesamiento del lenguaje natural, sería una alternativa cuestionable, ya que durante el desarrollo de la presente investigación se descubrieron casos de ambigüedad, los cuales, no reportaron error. En este escenario, el

procesamiento del lenguaje natural, sería difícil involucrarse, porque la ambigüedad semántica, a saber, es un problema que limita trabajar con este tipo de estrategias. Estos casos observados (de ambigüedad semántica en los textos cifrados), no se han reportado en la presente investigación, ya que, esta situación envuelve una gran variedad de explicaciones que podrían ser cubiertas en un trabajo futuro.

4. CONCLUSIONES Y/O PROYECTOS FUTUROS.

Es bien conocido, que el proceso de cifrado de datos de algunos algoritmos o métodos de encriptado estático (Barranco & Galindo, 2022; Courtois, 2012; Dang & Le, 2022; Dhany et al., 2018; Freda, 2022; Fulgueira et al., 2015; Gómez et al., 2012; Gómez & Díaz, 2021; Lake, 2023; Li & Wang, 2022; Lowery, 2023; Luciano & Prichett, 1987; Mendoza, 2008; Pieprzyk & Tombak, 1994; PortalCripto, 2023; Van & Jajodia, 2011; Van-Tilborg, 2005), es mucho más rápido que algunas propuestas dinámicas y/o basadas en algoritmos genéticos (Chong et al., 2011; Delman, 2004; Griindlingh & Van-Vuuren, 2002; Hossam et al., 2007;Jiménez et al., 2015; Kalsi et al., 2018; Matthews, 1993; Pisarchik & Zanin, 2008; Rangel et al., 2023; Rangel & Rangel, 2024; Reddaiah, 2016, 2019; Rodríguez, 2020; Wang & Chen, 2021). Sin embargo, elegir alguna de las alternativas estáticas, no es bueno para las organizaciones, debido a que, como no presentan variaciones en el resultado obtenido como texto cifrado, esta situación puede hacer que la seguridad de los datos sea vulnerable a ataques por parte de los ciber-delincuentes. En un trabajo de investigación propuesto por Rangel et al. (2023), se señala al respecto, que "si determinado lenguaje de programación, permite el cifrado, por ejemplo, del algoritmo MD5; solo bastaría «entrenar un modelo supervisado» durante un lapso largo de tiempo, para construir un diccionario con IA; ya sea, mediante procesamiento de lenguaje natural y/o métodos basados en semántica web, para poder descifrar, por ejemplo, un texto o clave de 16 caracteres". En cambio, los métodos de cifrado dinámico (Chong et al., 2011; Griindlingh & Van-Vuuren, 2002; Hossam et al., 2007; Jiménez et al., 2015; Kalsi et al., 2018; Matthews, 1993; Pisarchik & Zanin, 2008; Reddaiah, 2016, 2019; Rodríguez, 2020; Wang & Chen, 2021), particularmente, los basados en métodos aleatorios (Delman, 2004; Rangel et al., 2023; Rangel & Rangel, 2024), se puede observar que son alternativas más seguras para encriptado de la información, ya que, el texto cifrado obtenido, incluso con una misma cadena de texto plano de entrada, produce distinto resultado en cada ejecución o experimento (ver Tabla 3), y ello hace que la información sea menos vulnerable a ataques, por parte de los ciber-criminales.

Lo anterior, concierne con la presente investigación, porque en este trabajo se evalúan dos métodos de cifrado dinámico. Al respecto, se considera que los objetivos fueron alcanzados. En primer término, porque se hizo una modificación al procedimiento del algoritmo genético estudiado, denominado como: *Noised random pseudo-hexadecimal GAs*, y se logró reducir el tiempo empleado para el cifrado de datos (ver Tabla 2). En segundo término, se logró evitar el uso del procedimiento basado en algoritmo genético, evitando caer en un ciclo infinito. Este esquema, se observa en la Tabla 2, dónde ocurrió un error en el experimento #4. Detalles de este caso, ya han sido comentados anticipadamente. Dicha situación, ha dado lugar a la nueva variante denominada: *algoritmo aleatorio con ruido pseudo-hexadecimal: noised random pseudo-hexadecimal* (eliminando el calificativo GAs). Esta nueva propuesta, es recomendada como nueva alternativa en el cifrado de datos dinámico, ya que, se observó durante la experimentación, que este nuevo método permite alcanzar la convergencia en menos tiempo sin hacer uso del algoritmo genético, siendo capaz de producir distintos resultados, aún cuando se utilice la misma entrada de texto plano (ver Tabla 2 y Tabla 3). A manera de poder presentar un balance final sobre la investigación realizada, se puede observar en la Tabla 2, que la nueva propuesta: *noised random pseudo-hexadecimal*, es 23.19 veces más rápida para realizar el cifrado de datos, en comparación con: *Noised random pseudo-hexadecimal GAs*. Aunque la diferencia promedio de velocidades para el descifrado de datos, no es muy sustancial, ya que, la nueva propuesta solo es 1.14 más rápida que la alternativa basada en GAs. Sin embargo, debemos recordar, que los datos de entrada utilizados como texto plano, fueron empleados con una extensión menor o igual a 255 caracteres. En este contexto, se recomendaría realizar pruebas o experimentos con archivos en diferentes formatos, aunque ello demandaría la realización de algunas adaptaciones al procedimiento, lo cual, podría considerarse como un trabajo a futuro.

Con respecto a las dificultades observadas en el funcionamiento del algoritmo genético para el cifrado de datos dinámico, una de ellas, es que puede llegar a seleccionar valores fuera del rango de la tabla ASCII o algunos caracteres que no son imprimibles en pantalla, ello puede producir pérdida de información y reportar errores al momento del descrifrado de datos, lo cual, constituye una desventaja comparado con: *noised random pseudo-hexadecimal,* debido a que, la experimentación con este último esquema, no presentó dicho problema. Por último, si observamos resultados obtenidos por otros autores, podemos apreciar que Rangel & Rangel (en revisión desde 2024), exponen un método denominado: *noised random pseudo-hexadecimal*, pero se específica que fue implementado: "*por sustitución*". Empero, si observamos el procedimiento descrito en ese trabajo, es diferente, al empleado en la presente investigación. Dentro de este mismo contexto, la nueva propuesta aquí presentada como: *noised random pseudo-hexadecimal* es diferente a la alternativa: *noised random pseudo-hexadecimal (by substitution)* que se describe en Rangel & Rangel (en revisión desde 2024). Cabe aclarar que ambas propuestas persiguen evitar el uso del algoritmo genético para el cifrado de datos. Sin embargo, valdría la pena comparar ambas variantes, incluso se podría implementar otras modificaciones al procedimiento, ya que, no solamente podrían trabajar el cifrado de datos "por sustitución", sino también, "por desplazamiento" (o ambas vertientes). Este podría ser otro trabajo futuro a desarrollar, con la aplicación del formato *pseudo-hexadecimal*.

5. AGRADECIMIENTOS

Este trabajo fue financiado parcialmente por el Tecnológico Nacional de México, registrado con clave: 19329.24-P.

6. REFERENCIAS BIBLIOGRÁFICAS

Android 12 (2024). [Sistema operativo para dispositivos móviles]. Google. Recuperado de: https://www.android.com/intl/es_es/android-12/, (18/11/2024).

Al-Riyami, M., and Al-Khateeb, B. (2020). "Malicious Attacks Detection in Cloud Computing Using Machine Learning Techniques". Journal of Information Security and Cybercrime, 9(1), 1-11.

Al-Riyami, M., and Al-Khateeb, B. (2021). "Review of Data Decryption Tools and Techniques". International Journal of Advanced Science and Technology, 30(2), 555-564.

Álvarez, D. (2019). "Algunos Aspectos Jurídicos Del Cifrado De Comunicaciones". Derecho PUCP, núm. 83, 2019, pp. 241-264. Pontificia Universidad Católica del Perú. DOI: https://doi.org/10.18800/derechopucp.201902.008. Recuperado de: http://www.redalyc.org/articulo.oa?id=533662765008, (09/11/2024).

Anónimo (2021). "Tipos de algoritmos criptográficos: Cifrados de flujo". The Dojo. Recuperado de: https://blog.thedojo.mx/2021/12/12/tipos-de-algoritmos-criptograficos-cifrados-de-flujo.html, (10/11/2024).

American National Standards Institute. (1963). "American Standard Code for Information Interchange (ASCII)". ANSI X3.4-1963. New York: ANSI. Retrieved from: https://www.ascii-code.com/, (11/11/2024).

Arai, F., and Sato, T. (2021). "Robotics Research: Current and Future Directions". Springer.

Barranco, F., & Galindo, C. (2022). "Criptografía básica y algunas aplicaciones". Universidad Jaume I, Departamento de Matemáticas, Castellón, España. Recuperado de: https://repositori.uji.es/xmlui/bitstream/handle/10234/201359/TFM_2022_Barranco_Bl%C3%A1zquez_FranciscoMiguel.pdf?sequence=1, (09/11/2024).

Barker, E., & Roginsky, A. (2020). "Recommendation for key management: Part 1 - General". National Institute of Standards and Technology.

Bishop, C.M. (2022). "Pattern recognition and machine learning", (2ª ed.). Springer.

Bruzzone, L., and Serpico, S.B. (1997). "Classification of Imbalanced remote-sensing data by neural networks". Elsevier Science B.V. , 0167-8655, 97. PH S0167-8655 (97) 00109-8.

Chaudhary, R., and Sharma, S. (2020). "Analysis of Malicious Attacks on Web Applications". Journal of Cybersecurity, 1(1), 1-12.

Chaudhary, R., and Sharma, S. (2021). "Data Decryption Techniques: A Survey". International Journal of Computer Science and Information Security, 19(2), 1-10.

Chen, X., and Wang, J. (*2022*). "Quantum machine learning: A survey". Information Sciences, 593, 257-275. .

Chong F., Bib, L., Yu, S.M., Xiao, L., and Jun, J. (*2011*). "A Novel Chaos-based Bit level Permutation Scheme For Digital Imagen Encryption". Optics communications, volumen 284, august [en línea]. Retrieved from: www.elsevier.com/locate/optcom, (09/11/2024).

Clark, A. (1994). "Modern optimisation algorithms for cryptanalysis". In Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, November 29 December 2, (pp. 258-262).

Courtois, N.T. (*2012*). "Security Evaluation Of GOST 28147-89". In: View Of International Standardisation. Cryptologia, Vol. 36, no. 1, 2012, pp. 2-13.

Dang, Q. H., and Le, H. Q. (*2022*). "Improved cryptanalysis of the RSA algorithm using side-channel attacks". Journal of Information Security and Applications, 65, 103313.

Dhany, H.W., Izhari, F., Fahmi, H.,Tulus, M., and Sutarman, M. (*2018*). "Encryption and Decryption using Password Based Encryption, MD5, and DES". Published by Atlantis Press. ISSN: 2352-5398. Open Access: CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

Delman, B. (2004). "Genetic Algorithms in Cryptography". Thesis for the Degree of Master of Science in Computer Engineering. Rochester Institute of Technology (RIT Scholar Works). Department of Computer Engineering. Retrieved from: https://scholar.google.com.mx/scholar_url?url=https://repository.rit.edu/cgi/viewcontent.cgi%3Farticle%3D6460%26context%3Dtheses&hl=es&sa=X&ei=cbaZZoadNt246rQPnd604AU&scisig=AFWwaeaMfCM5ORUFQN6DU4LA3aEG&oi=scholarr, (11/11/2024).

Freda, A. (2022). "¿Qué es el algoritmo de hashing MD5 y cómo funciona?". Avast Academy. Recuperado de: https://www.avast.com/es-es/c-md5-hashing-algorithm#:~:text=El%20MD5%20%28algoritmo%20de%20resumen%20de%20mensajes%29%20es,persona%20a%20la%20que%20se%20lo%20ha%20enviado, (09/11/2024).

Fulgueira, M., Hernández, O.A., & Henry, V. (2015). "Paralelización Del Algoritmo Criptográfico GOST Empleando El Paradigma De Memoria Compartida". Lámpsakos, núm. 14, pp. 18-24. Fundación Universitaria Luis Amigó Medellín, Colombia. E-ISSN: 2145-4086; julio-diciembre. DOI: http://dx.doi.org/10.21501/21454086.1633. Recuperado de: http://www.redalyc.org/articulo.oa?id=613965326004, (09/11/2024).

Gómez, S., Arias, J.D. ; Agudelo, D. (2012). "Cripto-Análisis Sobre Métodos Clásicos De Cifrado". Scientia Et Technica, vol. XVII, núm. 50, abril, pp. 97-102. Universidad Tecnológica de Pereira Pereira, Colombia. ISSN 0122-1701 97. Recuperado de: http://www.redalyc.org/articulo.oa?id=84923878015, (09/11/2024).

Gómez, H., & Díaz, J. (2021). Análisis de algoritmos de cifrado simétricos y asimétricos para la protección de datos. Revista de Ingeniería y Tecnología, 20(2), 1-12.

Goodfellow, I., Bengio, Y., & Courville, A. (*2016*). "Deep learning". MIT Press.

Goodfellow, I., Bengio, Y., and Courville, A. (*2021*). "Deep learning". MIT Press.

Goodman, J. (1968). "Introduction To Fourier Optics". New York: McGraw-Hill.

Goyal, S., and Kaur, G. (*2020*). "Comparative Study of Malicious Attack Detection Techniques". International Journal of Computer Science and Information Security, 18(2), 1-10.

Goyal, S., and Kaur, G. (*2021*). "Comparative Study of Historical Ciphers: Caesar, Vigenère, and RSA". Journal of Cybersecurity, 1(1), 1-12.

Granados, G. (2006). "Introducción A La Criptografía". Revista Digital Universitaria, 7(7), s.p. Recuperado de: http://www.revista.unam.mx/vol.7/num7/art55/int55.htm, (09/11/2024).

Griindlingh, W., and Van-Vuuren, J. H. (*2002*). "Using Genetic Algorithms to Break a Simple Cryptographic Cipher". Retrieved March 31, 2003 from http://dip.sun.ac.za/~vuuren/abstracts/abstr_genetic.htm.

Han, J., Pei, J., and Kamber, M. (*2022*). "Data mining: Concepts and techniques", (4ª ed.). Morgan Kaufmann.

Hand, D., Mannila, H., and Smyth, P. (*2001*). "Principles of Data Mining". The MIT Press.

Hartmann, A.K. (*2020*). "Heuristic search in graphs". Journal of Artificial Intelligence Research, 68, 1-33.

Hossam, E.A., Hamdy, K., and Osama, S.F.A. (*2007*). "An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) For Image Encryption And Decryption". Informática, volumen 3, pp. 121-129.

Iyengar, S. S., Kannan, R., & Ganapathi, S. (*2021a*). "Inteligencia artificial y criptografía: Tendencias y desafíos". IEEE Transactions on Emerging Topics in Computing, 9(2), 833-844.

Iyengar, S. S., Kannan, R., & Ganapathi, S. (*2021b*). "Análisis de patrones en datos cifrados utilizando inteligencia artificial". Pattern Recognition Letters, 146, 168-175.

Javidi, B., and Horner, J.L. (1994). "Optical Pattern Recognition for Validation and Security Verification". Optical Engineering, 33, 1752-1756. DOI: https://doi.org/10.1117/12.170736.

Javidi, B., Zhang, G.S., and Li, J. (1997). "Encrypted Optical Memory Using Double-random Phase Encoding". Appl. Opt. 36, 1054-1058. Retrieved from: https://pubmed.ncbi.nlm.nih.gov/18250772/.

Jiménez, M., Flores, O., & González, M.G. (2015). "Sistema para codificar información implementando varias órbitas caóticas". Ingeniería. Investigación y Tecnología, vol. XVI, núm. 3, julio-septiembre, pp. 335-343. ISSN 1405-7743 FI-UNAM / ISSN: 1405-7743. Universidad Nacional Autónoma de México. Distrito Federal, México. Recuperado de: http://www.redalyc.org/articulo.oa?id=40440683002, (09/11/2024).

Johns, M.V. (*1961*). "An empirical Bayes approach to non-parametric two-way classification". In Studies in Item Analysis and Prediction, H. Solomon, Ed Stanford, Calif,: Standford University Press.

Kalsi, S., Kaur, H., and Chang, V. (*2018*). "DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation". Convergence of Deep Machine Learning and Nature Inspired Computing Paradigms for Medical Informatics. Image & Signal Processing; In Journal of Medical Systems, volume 42, Article number: 17. DOI: https://doi.org/10.1007/s10916-017-0851.

Kanal, L.N. (1963). "Statical methods for pattern classification". Philco Rept, originally appeared in T. Harley et al., Semi-automatic imagery screening research study and experimental investigation, Philco Reports, V043-2 and v043-3, Vol. I, sec. 6 and Appendix H, prepared for U.S. Army Electronics Research and Development Lab. Under Contract DA-36-039-sc-90742, March 29.

Katz, J., & Lindell, Y. (*2019*). "Introduction to modern cryptography". CRC Press.

KeepCoding (2023). "¿Qué es SHA-1?". Keepcoding.(https://keepcoding.io/blog/que-es-sha-1/, 09-11-2024).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (*2012*). "ImageNet classification with deep convolutional neural networks". Advances in Neural Information Processing Systems, 25.

Kumar, A., and Sharma, S. (*2021*). "A Study on Historical Cryptographic Techniques: Caesar Cipher to DES". International Journal of Advanced Science and Technology, 30(2), 555-564.

Kumar, P., and Singh, S. (*2022*). "Comparative Analysis of MD5 and SHA-256 Hash Functions". Journal of Advanced Research in Dynamical and Control Systems, 14(5), 1-10.

Kuncheva, L. I., and Jain, L. C. (1999). "Nearest Neighbor Classifier: Simultaneous editing and feature selection". Pattern Recognition Letters, 20, 1149-1156.

Lake, J. (2023). "The MD5 algorithm (with examples)". Comparitech. (https://www.comparitech.com/blog/information-security/md5-algorithm-with-examples/, 09-11-2024).

LeCun, Y., Bengio, Y., and Hinton, G. (*2015*). "Deep learning". Nature, 521(7553), 436-444.

Li, J., and Wang, X. (*2022*). "New Collision Attacks on SHA-1 Based on Differentiated Path Construction". Journal of Cryptology, 35(3), 1-25.

Li, M., Chen, X., and Wang, X. (*2022*). "Phase conjugation encryption in photorefractive crystal: Security analysis and improvement". Journal of Optics, 24(5), 055402.

Linfei, C., and Daomu, Z. (2005). "Optical Image Encryption Based On Fractional Wavelet Transform". Opt. Comm. Vol. 254, p.p. 361-367. Retrieved from: https://ui.adsabs.harvard.edu/abs/ 2005OptCo.254..361C/abstract.

Liu, L., and Zhang, Y. (*2021*). "Discrete chaotic cryptosystem based on logistic map and RSA algorithm". Journal of Information Security and Applications, 56, 102714.

Liu, X., and Wang, X. (*2021*). A Secure Data Encryption Scheme Based on Chaotic Map and DNA Computing. Journal of Cybersecurity, 1(1), 1-12.

Liu, X., and Wang, X. (*2022*). "Quantum cryptanalysis of lattice-based cryptographic protocols". Physical Review X, 12(2), 021004.

Liu, Y., and Chen, X. (*2021*). "An Efficient Public-Key Cryptography Scheme Based on Lattice Reduction". IEEE Transactions on Information Forensics and Security, 16, 239-250.

Liu, Y., and Chen, X. (*2022*). "Cryptanalysis of MD5 Hash Function using Genetic Algorithm". International Journal of Intelligent Information Systems, 11(2), 1-12.

Lowery, J. (2023). "MD5 vs SHA-1 vs SHA-2: ¿Cuál es el Hash cifrado más seguro y cómo verificarlos?". FreeCodeCamp.(https://www.freecodecamp.org/espanol/news/md5-vs-sha-1-vs-sha-2-cual-es-el-hash-cifrado-mas-seguro-y-como-verificarlos/, 09-11-2024).

Lowery, J., & Pereyra, E. (2023). "MD5 vs SHA-1 vs SHA-2: ¿Cuál es el Hash cifrado más seguro y cómo verificarlos?". FreeCodeCamp. (https://www.freecodecamp.org/espanol/news/md5-vs-sha-1-vs-sha-2-cual-es-el-hash-cifrado-mas-seguro-y-como-verificarlos/, 09-11-2024).

Luciano, D., and Prichett, G. (1987). "Cryptology: From Caesar Ciphers To Public-key Cryptosystems". The College Mathematics Journal, vol 18 pp 2-17. Retrieved from: http://www.jstor.org/stable/2686311.

Matthews, R.A.J. (*1993*). "The use of genetic algorithms in cryptanalysis". Cryptologia, 17(4), 187-201.

Mendoza, J.C. (2008). "Demostración De Cifrado Simetrico Y Asimétrico". Ingenius. Revista de Ciencia y Tecnología, núm. 3, pp. 46-53. Universidad Politécnica Salesian. Cuenca, Ecuador. ISSN: 1390-650X. Recuperado de: http://www.redalyc.org/articulo.oa?id=505554806007, (09/11/2024).

Microsoft (2024). "Descarga de software". Microsoft. Retrieved from: https://www.microsoft.com/es-mx/software-download, (18/11/2024).

Mitchell, T.M. (*2020*). "Machine learning", (2ª ed.). McGraw-Hill.

Mogensen, P., and Glückstad, J.A. (*2000a*). "Phase-based Optical Encryption System With Polarisation Encoding". Opt. Commun. pp. 173, 177-183

Mogensen, P., and Glückstad, J. (*2000b*). "Phase-only Optical Encryption". Opt. Lett. 25, 566-568.

Mogensen, P., and Glückstad, J. (*2001a*) "Phase-only optical decryption of a fixed mask". Appl. Opt. 40,1226-1235.

Mogensen, R. E., and Glückstad, J. (*2001b*). "High Capacity Optical Encryption System Using Ferro-Electric Spatial Light Modulators". J. Optics A. 3, 10-15.

Murphy, K. P. (*2022*). "Probabilistic machine learning: An introduction". MIT Press.

Nagaraj, S., and Srinadth, V. (*2015*). "Data Encryption and Authetication Using Public Key Approach". International Conference on Computer, Communication and Convergence (ICCC 2015).

Nils-Nilson (1965).."Learning Machines". New York: McGraw Hill, pp 120-121.

Nomura, T., and Javidi, B. (*2000*). "Optical Encryption System Using A Binary Key Code". Journal of Applied Optics, vol. 39, pp. 4783-4787, September 10.

Oppliger, R (*s.f*). Contemporary cryptography, 1ra. ed., Boston, Artech.

Patel, H., and Patel, R. (*2020*). "Malicious Attack Detection Using Deep Learning Techniques". Journal of Information Security and Cybercrime, 9(2), 1-11.

Pieprzyk, J., and Tombak, L. (*1994*). "Soviet Encryption Algorithm". University of Wollongong, Department of Computing Science.

Pisarchik, A.N., and Flores-Carmona, N.J. (*2006*). "Computer Algorithms For Direct Encryption And Decryption Of Digital Images For Secure Communication". Proceeding of the 6th WSEAS International Conference On Applied Computer Science (Canary Islands, Spain), pp. 29-34.

Pisarchik, A.N., and Zanin, M. (2008). "Imagen Encryption Witch Chaotically Coupled Chaotic Maps". Elsevier Physica, abril [en línea], D 237. Retrieved from: www.elsevier.com/locate/physd.

PortalCripto (2023). "Algoritmo Hash SHA-256: qué es y cómo funciona". PortalCripto.(https://portalcripto.com.br/es/DICIONARIO/Algoritmo-hash-sha-256-qu%C3%A9-es-y-c%C3%B3mo-funciona/, 09-11-2024).

Progress Software Corporation, Telerik (2022). "Cifrado Y Transferencia De Archivos: Los Mejores Cifrados Seguros Para La Transferencia De Archivos". Ipswitch Blogs. Recuperado de: https://ipswitch.com/amp/es/los-mejores-cifrados-seguros-para-la-transferencia-de-archivos/, (09/11/2024).

Pydroid3 versión 7.4_arm64 (2024). [IDE for Python 3. Lenguaje de programación y compilador]. Google Play Store. Retrieved from: https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&pli=1, (18/11/2024).

Python.org (2024). "The Python Network". Python.org. Retrieved from: https://www.python.org/downloads/, (18/11/2024).

Rahman, M. S., and Hossain, M. S. (*2021*). "A Secure Private Key Cryptography Scheme Using RSA and AES". Journal of Cybersecurity, 1(1), 1-9.

Rajan, B., and Saumitr, P.A. (2006). "Novel Compression And Encryption Scheme Using Variable Model Arithmetic Coding And Coupled Chaotic System". IEEE Transactions on circuits and system- I, april, vol. 53 (No. 4). Retrieved from: https://www.researchgate.net/publication/3451216_A_novel_compression_and_encryption_scheme_using_variable_model_arithmetic_coding_and_coupled_chaotic_system.

Rangel, E. (2002). "Vecinos Envolventes para Variantes de la Regla del Vecino más Cercano". MSc.Thesis, Instituto Tecnológico de Toluca, México.

Rangel, E. (2022). "La Regla De Los k Vecinos Más Cercanos (k-NN) Basada En Distancia De Manhattan (City-Block) Para Mejorar La Clasificación De Patrones". Ponencia presentada en: Quinto Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Tecnológico Nacional De México (campus: Instituto Tecnológico de Cd. Altamirano). Noviembre, 2022. Cd. Altamirano, Estado De Guerrero, México (Artículo En Extenso). Recuperado de: http://erangel.coolpage.biz/pappers/edgarrangel2022.pdf.

Rangel, E. (2024b). "Documento Entregable Que Corresponde Al Objetivo y Actividad #1". [Informe no publicado]. Primer Informe Parcial, Proyecto: (19329): La Regla Del Vecino Más Cercano Como Alternativa Para Inyectar Ruido A Mensajes Encriptados Por El Algoritmo: Noised Random Pseudo-Hexadecimal (Clave: 19329.24-P). Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Departamento de Sistemas y Computación (01/08/2024).

Rangel, E., and García, O. (2002). "Nearest Centroid Neighbour, An Alternative in the Speech Recognition for the Execution from a Mobile Robot Simulator (Applied to the Imbalanced Training Sample Problem)". In: 9th International Congress On Computer Science Research (CIICC 02). October 23-25 2002 - Puebla, México. ISBN: ISBN-970-18-8526-0.

Rangel, E., and Rangel, K.U. (2024). "Novel Random Encryption Methods Based On Mutation Strategies Of Artificial Intelligence". Scientific and Practical Cyber Security Journal (SPCSJ), Vol. 8, No. 3. September Issue 2024. ISSN: 2587-4667 (electronic). Edited by SCSA - SPCSJ - BOAI. Published by Scientific Cyber Security Association in Georgia. Tbilisi, Georgia. (https://journal.scsa.ge/issue/september-2024/, 09/11/2024).

Rangel, E., and Rangel, K.U. (en revisión). "Novel Pseudo-Hexadecimal Encryption Strategies For Camouflaging Ciphertext Based On Nearest Neighbor With Artificial Intelligence". International Journal of Combinatorial Optimization Problems and Informatics (IJCOPI). Editorial Académica Dragón Azteca, S. de R.L. C.V. ISSN: 2007-1558. México.

Rangel, E., Rangel, K.U., Medrano, J., Bernal, C.A., & González, L. (2023). "Algoritmo Genético Para Cifrado De Datos, Basado En Un Nuevo Concepto Pseudo-Hexadecimal Con Inteligencia Artificial". Aceptado en: Sexto (VI) Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas, (en prensa). Tecnológico Nacional De México, Instituto Tecnológico de Cd. Altamirano. Noviembre, 2023. Cd. Altamirano, Estado De Guerrero, México. Recuperado de: https://www.cdaltamirano.tecnm.mx/index.php/17-vi-congreso-nacional-de-investigacion-en-ciencia-e-innovacion-de-tecnologias-productivas/140-tecnm-40, (09/11/2024).

Reddaiah, B. (2016). "A Study on Pairing Functions for Cryptography". IJCA (0975-8887), Vol. 149, No. 10, September, pp.4-7.

Reddaiah, B. (2019). "A Study on Genetic Algorithms for Cryptography". International Journal of Computer Applications (0975 – 8887). Volume 177 - No. 28, December. Department of Computer Applications. Yogi Vemana University Kadapa, A.P, India. Retrieved from: https://www.researchgate.net/publication/338012809_A_Study_on_Genetic_Algorithms_for_Cryptography.

Rodríguez, J. (2020). "Operadores Genéticos Aplicados A La Criptografía Simétrica". Proyecto De Grado. Universidad Distrital Francisco José De Caldas. Facultad De Ingeniería. Ingeniería De Sistemas. Bogotá, Colombia. Recuperado de: https://repository.udistrital.edu.co/handle/11349/28192.

Ross-Quinlan, J. (1993). "C4.5: Programs for Machine Learning". Morgan Kaufmann, San Mateo, CA.

Rueda, A.S., & Lasprilla, M. (*2002*). "Encriptación Por Conjugación De Fase En Un BSO Utilizando Señales Ópticas De Baja Potencia". Rev. Col. Fís., Vol. 34, No.2, (2002), P.P.636-640.

Rueda, J.E., Romero, A.L., and Castro, L.M. (2005). "Criptografía Digital Basada En Tecnología Óptica". Bistua: Revista de la Facultad de Ciencias Básicas, vol. 3, núm. 2, julio, pp. 19-25. ISSN 0120 - 4211. Universidad de Pamplona, Colombia. Recuperado de: http://www.redalyc.org/articulo.oa?id=90330203, (09/11/2024).

Russell, S.J., & Norvig, P. (*2020*). "Inteligencia artificial: Un enfoque moderno", (4ª ed.). Pearson.

Saini, H., and Gupta, S. (*2021*). "Private Key Cryptography Based Secure Data Storage and Retrieval in Cloud Computing". International Journal of Cloud Computing, 10(1), 1-15.

Sánchez, J.S., Pla, F., and Ferri, F.J. (1997b). "Prototype selection for the nearest neighbor rule through proximity graphs". Pattern Recognition Letters 18, 507-513.

Sebas, C. (2023). "¿Qué son los Algoritmos Genéticos en las Inteligencias Artificiales?". Manuales y Tutoriales de Informatica. Recuperado de: https://aprendeinformaticas.com/ia/, (11/11/2024)

Sebestyen, G. (*1962*). "Decision Making". Process in Pattern Recognition. New York: Macmillan, pp. 90-92.

Siegwart, R., and Nourbakhsh, I. R. (*2011*). "Introduction to autonomous mobile robots", (2ª ed.). MIT Press.

Singh, A.K., Kumar, P., & Singh, R. (*2021*). "Aplicación de la inteligencia artificial en la criptografía: Una revisión". Journal of Intelligent Information Systems, 67(2), 257-275.

Singh, P., and Kumar, P. (*2022a*). "Optical encryption using fractional Fourier transform and spiral phase plate". Optics Communications, 503, 127629.

Singh, P., and Kumar, P. (*2021*). "Brute force attack on password authentication: A review". International Journal of Advanced Research in Computer Science, 12(1), 1-9.

Singh, R., and Kaur, G. (*2021*). "Evolution of Cryptography: From Ancient Ciphers to Modern Cryptographic Techniques". Journal of Information Security and Cybercrime, 10(1), 1-11.

Singh, R., and Kaur, G. (*2021b*). "Comparative Analysis of Data Decryption Tools". Journal of Information Security and Cybercrime, 10(1), 1-11.

Singh, S. (2000). "Los códigos secretos". Madrid: Debate.

Skalak, D. B. (1994). "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms". In: Proceedings of the Eleventh International Conference on Machine Learning (ML94). Morgan Kaufmann, pp. 293-301.

Stinson, D. R. (*2022*). "Cryptography: Theory and Practice". (4ta ed.). Chapman and Hall/CRC.

Tajahuerce, E., and Javidi, B. (*2000a*). "Encrypting Three Dimensional Information With Digital Holography". Journal of Applied Optics, vol. 39, pp. 6595-6601, December 10.

Tajahuerce, E., Matoba, O., Verrall, SC., and Javidi, B. (*2000b*). "Optoelectronic Information Encryption Using Phase-shift Interferometry". Journal of Applied Optics, Vol. 39, pp. 2313-2320, May 10.

Tajahuerce, E., Lancis, J., Javidi, B., and Andrés, P. (*2001a*). "Optical Security And Encryption With Totally Incoherent Light". Journal of Optics Letters, vol. 26, pp. 678-681, May 15.

Tan, X , Matoba, O., Shimura, T., Kuroda, K., and Javidi, B. (*2000*). "Secure Optical Storage That Uses Fully Phase Encryption". Journal of Applied Optics,.vol. 39, pp. 6689-6694, December 10.

Tan, P.N., Steinbach, M., and Kumar, V. (*2022*). "Introduction to data mining", (2ª ed.). Pearson.

Thakur, J., Kumar, N. (*2011*). "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis". International Journal of Emerging Technology and Advanced Engineering, 6-12.

Unicode Consortium (2022). "The Unicode Standard, Version 14.0". Unicode Consortium. Retrieved from: https://www.unicode.org/consortium/consort.html, (11/11/2024).

Van, H.C., and Jajodia, S. (2011). "Encyclopedia Of Cryptography And Security". Springer Science & Business Media, 2011. 1416p. ISBN: 978-14419-5907-2.

Van-Tilborg, H.C.A. (2005). "Encyclopedia Of Cryptography And Security". pp 114-115, 201-202. TUE Research portal. Springer. https://doi.org/10.1007/0-387-23483-7, (09/11/2024).

Wang, J., and Zhang, Y. (*2021*). "Public-Key Cryptography with Lattice-Based Cryptography". Journal of Cryptology, 34(2), 1-25.

Wang, X., and Chen, G. (*2021*). "Discrete-time chaotic cryptosystem based on memristive neural networks". IEEE Transactions on Neural Networks and Learning Systems, 32(5), 2223-2234.

Wang, X., and Chen, X. (*2022*). "Optical image encryption based on compressive sensing and double random phase encoding". Journal of Optics, 24(2), 025401.

William, S. (1999). "Cryptography and Network Security: Principles and Practice". 2nd edition, Prentice-Hall, Inc., pp 23-50. Retrieved from: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.cs.vsb.cz/ochodkova/courses/kpb/cryptography-and-network-security_-principles-and-practice-7th-global-edition.pdf&ved=2ahUKEwjXsIql8rGHAxWsKUQIHTA-APIQFnoECBQQAQ&usg=AOvVaw2IROGmmRSXMbAjzdVHzwug.

Zhang, Y., and Yang, Q. (*2022*). "Deep learning for data mining: A survey". ACM Computing Surveys, 54(2), 1-37.