Intelética 1(2) (2024), 1-15
Revista de Inteligencia Artificial, Ética y Sociedad

INTELETICA

https://inteletica.iberamia.org/

La regla del vecino más cercano como alternativa para inyectar ruido a mensajes encriptados por el algoritmo: Noised Random Hexadecimal

Edgar Rangel-Lugo [1], Kevin Uriel Rangel-Ríos [2]

[1] Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano.
[2] Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano.

[1] erangel_lugo@hotmail.com
[2] kgvppro@gmail.com

Abstract The theft of digital data problem is receiving growing attention. This situation can arise if at least one of the cybersecurity strategies are not updated periodically (e.g., the employment of inadequate or static encryption methods). It has been observed in several practical domains that may produce an important losses in the organisation's finances. In the present paper, several aspects related with the subject are studied and experimental results are here shown. We recommend the replacement of static encryption method by another dynamic alternative. Moreover, simultaneously was employing a noisy injection to the ciphertext previously created by the static strategies instanced in the organisations. The advantage of a dynamic encryption methods is creating a different ciphertext result after of each execution, with the same plain text input. Novel proposal based on nearest neighbor (1-NN) for ciphering of the information dynamically with noisy injection in hexadecimal format and comparison results with traditional strategies (AES, RSA, 3DES, RC4 and DES) are also introduced.

Resumen El problema de robo digital de datos está recibiendo gran atención. Esta situación puede presentarse cuando una de las estrategias de ciberseguridad no es actualizada periódicamente (por ejemplo, el uso de algoritmos de cifrado obsoletos o estáticos). Ello se ha observado en varios dominios prácticos, que puede producir importantes pérdidas en las finanzas de las organizaciones. En el presente artículo, son estudiados varios aspectos relacionados con dicha área. Además, se muestran resultados experimentales que nos permiten recomendar una alternativa para reemplazar el método de encriptado de datos estático, por una variante dinámica que puede inyectar ruido a mensajes cifrados por el algoritmo obsoleto o estático que actualmente se utiliza en las organizaciones. Un método de cifrado dinámico, permite generar diferentes resultados, en cada ejecución, para una misma secuencia de entrada de texto plano. Por último, en este trabajo se introduce una nueva propuesta sobre el uso de la regla del vecino más cercano (1-NN) para el cifrado de datos dinámico con inyección de ruido en formato hexadecimal y se comparan resultados con cinco algoritmos estándar (AES, RSA, 3DES, RC4 y DES).

Palabras clave: Metodología Random Caesar, criptografía, la regla del vecino más cercano, formato hexadecimal.
Keywords: Random Caesar methodology, cryptography, nearest neighbor rule, hexadecimal numbers.

1 Introducción

El problema de robo de datos digitales, en las organizaciones esta recibiendo gran atención. En la práctica se han observado grandes pérdidas económicas cuando se presenta una situación de dicha índole, particularmente en instituciones que no cuentan con algún mecanismo de seguridad implementado. Una alternativa muy popular para

ISSN: 3020-7444

ayudar a ese problema, es conocida como cifrado de datos, cuyo estudio corresponde a una de las área de las ciencias computacionales, dentro del campo de la seguridad informática, ya que, existe una área conocida como ciberseguridad [1]-[5] que en su nueva modalidad de actualización, recibe el nombre de ciber-resiliencia. En dichas áreas, ha preocupado el aspecto de mantener a salvo la información de las organizaciones, así como de usuarios particulares, que manejan grandes volúmenes de datos. En dichos campos de estudio se encuentra ubicada la criptografía, que proporciona herramientas necesarias para el cifrado de datos. La criptografía ha sido usada casi simultáneamente desde el desarrollo avanzado del lenguaje escrito [6] y tradicionalmente jugó un rol fundamental en la protección de las comunicaciones oficiales de los Estados, de los gobernantes y, principalmente, de las instituciones militares [7]. El uso de criptografía, cifrado y descifrado de texto, fue iniciado alrededor de 1900 B.C.; cuando los Egipcios comenzaron a aplicar procedimientos de correspondencia [4],[8]. Se entiende por cifrado de datos, 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. En actualidad, existen procesos dónde se usa la Inteligencia Artificial (IA) para conseguirlo. Uno de los propósito de la IA es hacer que "la máquina piense" [9]-[13]. Para lograr este propósito, la IA se apoya en diversas estrategias, técnicas, arquitecturas, modelos y paradigmas; algunos de ellos, basados en métodos estadísticos [14], métodos heurísticos, por ejemplo: los árboles de decisión [15] o gráfos [16]; los métodos aleatorios [18]-[19], entre otros modelos. Un trabajo reciente, que utiliza IA, mediante métodos supervisados, para el encriptado de datos [20], sugiere el uso de un algoritmo genético [4], [17]-[18] para lograr el cifrado de datos con ruido, usando un formato denominado: Pseudo-Hexadecimal [20]. Un método supervisado [9],[21], es aquel que aprende a partir de una muestra de entrenamiento (ME). Todo método supervisado consta de dos etapas: El aprendizaje y la producción [9]-[13], [21]. Existen una gran variedad de métodos supervisados, por ejemplo: árboles de decisión basados en el algoritmo C4.5 [15], la Regla NN (por sus siglas en inglés: Nearest Neighbor), mejor conocida como: 1-NN o regla del vecino más cercano [22], las redes neuronales supervisadas [23] entre otros. En la etapa de aprendizaje, casi todos los métodos (excepto la regla NN), entrenan el modelo usando la muestra de entrenamiento (ME). Después, se desecha y se procede a trabajar con clasificación [9]. Entonces, las redes neuronales, utiliza su modelo matemático para modificar los valores de sus enlaces, llamados "pesos" durante el aprendizaje, y en la producción, usa los pesos o modelo matemático para clasificar y permitir la toma de decisiones. Empero, la regla del vecino más cercano, no es obligatoria la etapa de aprendizaje [15], [24], que consiste en la creación y/o preprocesamiento de la muestra de entrenamiento; para editar, reducir, descontaminar, por mencionar algunas metodologías; y la etapa de producción, se usa la muestra de entrenamiento para la clasificación de patrones.

Por otra parte, existen otros trabajos relacionados con criptografía donde se utiliza la regla 1-NN y/o algoritmos genéticos (GAs). Por ejemplo, se han aplicado con éxito en la optimización y planificación de rutas de transporte de redes de energía y detección de fraudes en el sector financiero, robótica, diseño de circuitos electrónicos, aprendizaje automático [5]; así como, en criptografía con IA para comercio electrónico [4], en lo relacionado con ataques sobre la transposición y permutación de cifrado [1], en algunos casos, aplicado a mono-alfabetos. Además, existen otras alternativas [25] que proponen el uso de un algoritmo criptográfico simétrico aplicando GAs, Entropía y Aritmética Modular, mostrando comparación de resultados con: DES: Data Encryption Standard, RSA:Rivest, Shamir and Adleman y AES: Advanced Encryption Standard [25]. Del mismo modo, podemos encontrar en la literatura, información acerca de otros tipos de métodos para cifrado de datos [26], basados en encriptación óptica de información que usa mascaras aleatorios de fase; mientras que en otras propuestas [27]-[28] se utiliza una criptografía digital basada en tecnología óptica. Algunos trabajos en este sentido, que usan llaves ópticas aleatorias de fase en el plano de entrada y en el plano de Fourier, son los reportados en dichas investigaciones [26]-[27], incluyendo una que recientemente presentó un método que usa la transformada wavelet [29]; aunque muchos métodos o esquemas de comunicación segura se han desarrollado también, para cifrar información basándose en sistemas discretos caóticos [30]-[33]. Por otra parte, existe otro trabajo [2] que muestra una visión de algoritmos simétricos y asimétricos; aplicando ambas técnicas mediante el uso de RSA y 3DES, implementados en Visual Basic. NET. También, existen los algoritmos de cifrado simétrico, discreto caótico [34], de mapa logístico. Este algoritmo se aplicó para codificar una imagen. Otro algoritmo criptográfico, que presenta un proceso de paralelización, es el algoritmo criptográfico GOST [35], utilizado para reducción del tiempo de ejecución de un algoritmo criptográfico; mientras que en otras investigaciones se exponen algunos aspectos jurídicos del cifrado de comunicaciones [7].

En lo relacionado con los algoritmos de cifrado; en el pasado, Julio César (Julius Caesar)[36]-[37], también diseñó métodos seguros para transmitir en secreto información para gente importante en el campo de la milicia [4],[38].

De acuerdo con lo anterior, otros autores [36] presentan varios tipos de algoritmos de cifrado: por desplazamiento, por sustitución, por permutación, cifrados de flujo, de claves privadas (como: DES, AES), de clave pública (como: RSA), basados en funciones Hash (como: MD5, SHA1 y derivados), por mencionar algunos. Sin embargo, uno de los algoritmos más populares en sus inicios, por ser sencillo de implementar [20], fue el algoritmo de desplazamiento o sustitución, conocido como: Algoritmo Caesar o Cifrado del César [36]-[37]; el cual, tiene algunas variantes o derivados, que han desarrollado distintos investigadores, donde se muestran algunas técnicas de encriptación clásicas, tal como los cifrados del Cesar y Vigenère [37]; realizando un cripto-análisis sobre métodos clásicos de cifrado. Sin embargo, al observar en la literatura, podemos darnos cuenta que, los cifrados basados en Caesar, ya no son muy utilizados frecuentemente; destacando otras series populares como: MD5, SHA1 y derivados basados en tablas Hash, principalmente para tareas de autenticación o integridad de datos; entre otros algoritmos populares, empleados para el cifrado y descrifrado de información, como son el caso de: AES, DES, RSA, por mencionar algunos [39].

Empero, aunque existe gran variedad de algoritmos de cifrado [19], [36]; en la práctica, los usuarios de Internet, siguen estando vulnerables a ataques por parte de los ciberdelincuentes [20], ya que, algunos lenguajes de programación soportan librerías para cifrado/descifrado de datos, lo cual, les permite a estos ciberdelincuentes obtener información de manera rápida y fácil. Por tal razón, es importante que nos preocupemos por proteger los datos usando nuevos algoritmos o métodos, incluyendo el uso de cifrado de la información, sobretodo utilizar metodologías que no conozcan o soporten las herramientas libres o comerciales o "ilegales", que existen para el descifrado de datos. Es por ello, que este trabajo de investigación, se considera importante realizar la exploración del problema de inseguridad de datos, presentando algunas alternativas para el cifrado de información, que permita confundir y/o retrasar, a estos ciberdelincuentes, en el momento de querer descifrar nuestra información, y por ende, pueda mejorar la seguridad de datos en las organizaciones. Esta es precisamente, una de las hipótesis que guía este trabajo, que ¿A través de la inyección de ruido en un mensaje cifrado, puede incrementarse el grado de seguridad, por ejemplo, en una cadena de texto cifrada?. Una aproximación para resolver dicho problema, la podemos encontrar en otras investigaciones [2], [20], [29]-[33],[39].

En lo que concierne con la presente investigación, se propone el uso de inyección de ruido, en formato hexadecimal, introduciendo el nuevo concepto del uso de la regla del vecino más cercano [13],[22], para inyectar ruido en mensajes cifrados, todo con el propósito de generar un paquete encriptado de mayor dificultad para ser descifrado. Se comparan cuatro algoritmos, basados en el Cifrado del Caesar Aleatorio (Random Caesar [20]) con dos variantes del clásico Cifrado del César. Los métodos de cifrado estudiados son: Cifrado Del César por desplazamiento y por sustitución [36]-[37]; así como, las nuevas variantes propuestas: Hexadecimal Caesar, Hexadecimal Random Caesar, Noised Random Hexadecimal, y por su puesto, Noised Random 1-NN Hexadecimal. Adicionalmente, con el propósito de *validar las técnicas en entornos reales con datos diversos y poder comparar su efectividad respecto a otros métodos más tradicionales, se incluyen resultados de experimentos utilizando los algoritmos:* AES (Advanced Encryption Standard [25], [36], [39]), RSA (Rivest-Shamir-Adleman [2],[25],[36],[39]), 3DES (Triple Data Encryption Standard [2]), RC (Rivest Cipher [2]) y DES (Data Encryption Standard [2],[25],[36],[39]).

2 Metodología

El tipo de investigación desarrollada en el presente trabajo, se considera experimental y exploratoria, teniendo en cuenta que el algoritmo del César y variantes [36]-[37], [20], combinado con el uso de la regla 1-NN [22], [9]-[13], en el cifrado de datos, ha sido poco estudiado [20] o no se había utilizado antes del desarrollo del presente proyecto, o al menos no con el mismo enfoque o propósito que aquí se presenta. Los datos utilizados son del tipo número entero, hexadecimales y cadenas de caracteres (para su procesamiento, en algunos casos se utilizaron estructuras de vectores, en su formato de arreglos computacionales). Se trata de mensajes de texto que fueron convertidos en vectores (de caracteres ASCII, o su correspondiente ordinal entero y/o hexadecimal, según sea el caso). La recolección de datos fue llevada a cabo en forma heurística y de manera aleatoria. Primero, se diseñaron cadenas de texto a cifrar, usando una estructura de árbol, para distinguir texto en dos idiomas: español e inglés; incluyendo (en las secuencias de texto) algunos caracteres fuera del rango imprimible del código ASCII. Posteriormente, con esa información, se ha creado una muestra de entrenamiento (ME), que es una matriz generada en forma aleatoria con reemplazo, donde cada fila o patrón de entrenamiento, contiene el texto cifrado, cuya etiqueta de clase corresponde al mismo mensaje de texto original (el diseñado antes de ser cifrado). El tamaño máximo utilizado en las cadenas de texto a cifrar (columnas en la ME) fue de 255 caracteres (y por ende, las posiciones de los vectores de

procesamiento, también es de 255 como máximo; aclarando que cada posición del vector puede contener un valor ASCII de un solo caracter; o si es de tipo entero puede almacenar un valor entre 0 a 255. Del mismo modo, los vectores hexadecimales almacenan un valor de cuatro cifras, su correspondiente hexadecimal ASCII; es decir, la secuencia FF se almacena como 00FF). Cabe mencionar, que a cada método de cifrado utilizado, se diseñó su propia ME y fueron evaluados cada uno de manera separada. El procedimiento empleado para medir el acierto o llevar a cabo la estimación del error, a cada una de las muestras, se le aplicó validación cruzada con cinco repeticiones (el 20% empleado como muestra de control y un 80% para evaluar el modelo, se usa como muestra de entrenamiento). Además de las muestras de entrenamiento, se utilizaron otros materiales durante el proceso de desarrollo y experimentación. Se trata de dos equipos de cómputo con sistema operativo Windows 10, velocidad de procesamiento (CPU) de 2 GHz, capacidad de 8 GB en memoria RAM y un disco duro de 500 GB. Se instaló lenguaje Java (Oracle JDK versión 21), y también, lenguaje Python 3, para la implementación del software que realiza las operaciones de cifrado/descifrado de datos; el uso de dos lenguajes de programación, fue considerado para llevar a cabo comparaciones de los resultados (textos cifrados), así como, de los tiempos de ejecución, y poder verificar el grado de confiabilidad de los métodos de cifrado, que se describen más adelante. Al final, se hizo una prueba comparativa entre los resultados y/o código fuente generados, y no se observó diferencia significativa. Los algoritmos implementados en Python, funcionaron en las mismas condiciones y con comportamientos equivalentes respecto a los codificados en Java. Solamente se observó, que era mejor iniciar la implementación primero en Java, y después migrar el código a Python; para evitar demoras con el grupo de programadores, ya que, se detectaron partes de código escrito en Java, que no podían traducirse a Python del mismo modo, tal es el caso de: abstracción, uso de interfaces (implements), sobrecarga de métodos, polimorfismo, uso de ciclos "do-while", entre otras operaciones; que al ser migradas de Java a Python, en algunos casos, se tenía que codificar distinto, y por ende, se tuvo que regresar a convertir en Java su equivalencia, y ello demoraba el proceso de implementación. Los métodos empleados (e implementados en lenguajes de programación: Java y Python 3) son los que se describen enseguida.

El primer método de cifrado, utilizado en la presente investigación, es el clásico Cifrado del César, conocido también como Cifrado Caesar. Este algoritmo, es considerado de fácil implementación, que fue utilizado por El César, para comunicarse con sus generales [37]. Se trata de un tipo de cifrado por sustitución, donde un símbolo de texto plano (a cifrar) es sustituido por otro símbolo que se encuentra K posiciones siguientes en el mismo alfabeto. Por ejemplo, suponiendo que el valor de K=1 y el alfabeto es el abecedario del español y el texto plano es: "holaquetal", se procede a reemplezar cada letra del texto plano, por la siguiente del alfabeto (puesto que k=1), teniendo como resultado, que la letra "h" se sustituye por "i", la letra "o" se reemplaza por "p", y así sucesivamente, hasta agotar las letras que contiene el texto plano, obteniendo la secuencia cifrada o encriptada: "ipmbrvfubm" [36]-[37]. Una definición formal para lograr el cifrado con este método es la siguiente: $C[i] = S[i] + K \pmod N$; y el descifrado se utiliza: $D[i] = C[i] - K \pmod N$. Donde: S[i] es el carácter (en posición i) del texto plano; C[i] es el carácter i del texto cifrado; D[i] es el carácter i del texto descifrado; el parámetro K es el número de posiciones siguientes para poder calcular la operación de sustitución; mientras que N corresponde a la cantidad de símbolos del alfabeto; que se refiere al módulo 26, es decir, el número de letras en alfabeto [36]; por ejemplo, iniciando desde la letra "A" y terminando hasta la letra "Z", son 26 caracteres, si se eliminan del alfabeto español las siguientes letras: Ñ, CH, RR y LL. Por lo tanto, si se utiliza este método con el valor K=3, se conoce como Cifrado del César, debido a su uso reportado por Julio César [36]. El problema que tiene este algoritmo, es que hace vulnerable la seguridad de los datos; ya que, un mismo símbolo del texto plano, se puede cifrar al mismo símbolo del texto cifrado. Por lo tanto, si un criptoanalista conoce que para una secuencia cifrada se utilizó este método, podría localizar la letra que más se repite en el texto cifrado, por ejemplo, si se observara que la letra "e" del alfabeto se repite más, podría realizar la resta entre esas dos letras y hallar fácilmente el valor K para la sustitución [20], [37], [40]. Sin embargo, si observamos en la literatura [36]-[37], podemos apreciar que existen varios métodos de cifrado, por ejemplo: por desplazamiento y por sustitución, entre otros. El clásico Cifrado del César, algunos autores [37] lo describen como un tipo de cifrado por sustitución; mientras que en otras investigaciones [36] se describe como un algoritmo por desplazamiento. En la presente investigación, al clásico Cifrado del Caesar, se presenta como algoritmo de desplazamiento, ya que, de esa manera ha sido su implementación (en Java y Python 3). En esas referencias [36]-[37], no se describe el uso de caracteres fuera del alfabeto del módulo 26 (mod 26); por ejemplo: números y otros símbolos de la tabla ASCII, son omitidos. Tampoco, se menciona si se debe distinguir o no, el uso de minúsculas o mayúsculas. Por lo tanto, para poder comparar resultados con las fuentes citadas, también se hizo la implementación "por sustitución", que se refiere en este artículo como: Cifrado Del César por sustitución, que fue el segundo método de cifrado utilizado, aclarando que se aplicaron modificaciones en la implementación, para que "realmente" se trabajara "por sustitución", algo parecido a las ecuaciones que presenta el "Criptosistema 3" [36]. Por lo tanto, en la

presente investigación, la implementación del Cifrado del Caesar por sustitución, para obtener la secuencia cifrada, se puede definir formalmente como: C[i] = Z[i] (mod 26). Donde: Z[i] se obtiene como: Z[i]=D[t], solamente si: S[i]=A[t] (distinguiendo en el alfabeto A[t] las mayúsculas y minúsculas); en otro caso se asigna: Z[i]=S[i] (es decir, conserva el caracter original de la ocurrencia actual en S[i]). Cabe aclarar que el alfabeto A[t] tiene una extensión de "mod 26", pero ahora se distinguen las mayúsculas y minúsculas. La variable i avanza en función del tamaño de la secuencia S[i] (del texto plano); mientras que la variable t avanza en función del alfabeto A[t] o D[t], que en este caso su valor mayor es 26, por usar un módulo N=26. El procedimiento se realiza por sustitución del caracter localizado en alfabeto A[t] siendo reemplazado por el caracter del segundo alfabeto D[t] que está ubicado en la misma posición de A[t], para finalmente ser guardado en Z[i]. Es por ello, que en esta variante "por sustitución", permite la incorporación de caracteres que no estén dentro del rango de mod 26 (copiando el mismo caracter S[i] dentro de la secuencia de texto cifrado C[i]). Otra aclaración al respecto, es el uso de D[t], que se trata de un segundo alfabeto para realizar el cifrado/descifrado de datos, el cual, consiste en un alfabeto con desplazamiento de K, cuyo valor debe ser menor o igual que 26. Para obtener este segundo alfabeto se realiza la operación: D[t]=A[t+K]; si el valor de (t+K) es mayor que mod 26, se regresa a la posición inicial de A[t], y desde ahí, continúa trabajando como si se tratara de una estructura de cola circular. En esta investigación, un valor de K mayor que 26, se ajusta como K=26, lo que permite darnos cuenta de algún error en el cifrado de datos, debido a que, el resultado de C[i] corresponde a la misma información guardada en S[i], esto es, en caso de ocurrir error al localizar un caracter dentro del alfabeto A[t] o D[t], según sea el caso. Por último, para obtener el descifrado, se puede definir como: R[i]=Z[i] (mod 26); es decir, solo se intercambian los alfabetos: A[t] por D[t], sustituyendo S[i] por C[i], despejando del siguiente modo: S[i]=C[i]; Q[t]=A[t]; A[t]=D[t]; D[t]=Q[t]; posteriormente, se procede realizando las mismas operaciones del proceso de cifrado, obteniendo en R[i], el texto descrifrado.

El tercer método de cifrado utilizado en esta investigación, referido como: Hexadecimal Caesar (Cifrado Caesar Hexadecimal), cuyo procedimiento, es un derivado del Cifrado del César, implementado con la variante "por desplazamiento". En este aspecto, existe una diferencia, ya que, ahora se omite el uso del alfabeto, y en su lugar, se procede sumando el valor K (de desplazamiento) utilizando directamente los valores de la tabla ASCII del texto plano S[i], pero operando con el número correspondiente en hexadecimal. El resultado será una cadena concatenada de números en formato hexadecimal. Para obtener el cifrado usando Hexadecimal Caesar, se puede definir formalmente como: C[i]=(String) Hex((Ord(S[i]) + K) , Num) (mod 95); y para obtener el descifrado se puede utilizar: D[i]=Hex((Ord(C[i]) - K , Num) (mod 95). Donde: S[i] es el texto plano a cifrar; C[i] contiene los datos cifrados; D[i] contiene los datos descifrados; mientras que Num se refiere a las cifras hexadecimales a emplear (por ejemplo, para Num=2 y ASCII=255 indica el hexadecimal FF, si Num=4 el hexadecimal sería: 00FF). El valor de K indica el desplazamiento que debe ser menor o igual a 129 (de lo contrario, se ajusta como: K=129). El módulo 95 o mod 95, significa la extensión del alfabeto, pero no en términos de utilizar un vector para realizar desplazamiento (porque se omite), sino que, ahora se refiere a los caracteres ASCII válidos a considerar (se usan 95 caracteres de la tabla ASCII, del 32 al 126, que inicia con el espacio y termina con la tilde). Suponiendo que el mayor valor ordinal de S[i] fuera 126, al ser sumado con el máximo valor de K=129; tenemos que: 126+129=255, es decir, no excede el valor 255 de la tabla ASCII. Los valores en S[i] fuera del rango del mod 95, no representan problema, ya que, en ningún momento se realiza una conversión ASCII de un valor hexadecimal que ha sido aplicado previamente algún desplazamiento de K, solo que en este último caso, se recomienda trabajar conversiones de hexadecimales con Num=4. Por último, el uso de las funciones o "cast": Ord, Hex y String ; permiten convertir valores ASCII a ordinales, enteros a hexadecimales; y , de hexadecimales a cadena de caracteres; respectivamente. En esta investigación, se ha utilizado cuatro cifras para operaciones en hexadecimal, para permitir el uso de caracteres extendidos de la tabla ASCII, que al ser convertidos a ordinal, el lenguaje de programación puede regresar un entero fuera del mod 255 del ASCII (por ejemplo, el ASCII=178 es un caracter extendido imprimible como ' ▒ ' ; pero al ser convertido en hexadecimal, se observó en plataforma Windows, que no se traduce como: ASCII=178, sino que regresa un número muy grande, por ejemplo: 9714, cuyo valor hexadecimal rebasa las dos cifras: 25F2).

El cuarto método de cifrado utilizado en esta investigación, es denominado Hexadecimal Random Caesar (Cifrado Caesar Aleatorio Hexadecimal), el cual, fue inspirado en una de las versiones de Random Caesar (Cifrado del César Aleatorio), que han sido presentadas en otras investigaciones [20], debido a que, dichas propuestas resultaban muy prometedoras en el aspecto de incrementar la seguridad de los mensajes cifrados, presentando una nueva aportación al clásico Cifrado del César (por desplazamiento), a través de métodos aleatorios aplicados sobre el desplazamiento K, denominado ahora K[i], por la razón de que, en esta nueva modificación son varios desplazamientos que se realizan, cada uno de ellos (aleatorio con reemplazo) por caracter incluido en el texto plano S[i]; además de contar con una segunda fase en el procedimiento, que consiste en crear un empaquetado con inyección de ruido ordinal o tipo ASCII, existen al menos tres versiones, que difieren en la extensión del módulo N (comúnmente, N=255 versión

estándar, N=95 segunda versión con rango ASCII de 32 hasta 126; y , N=120, la versión II extendida con rango ASCII de 30 a 150, para inyectar ruido). Esta modificación, se considera que puede ayudar un poco a impedir que un "cyber-delincuente", logre descifrar fácilmente el texto C[i], ya que, para conseguirlo tendría prácticamente que "adivinar" cuál valor de desplazamiento se ha utilizado en cada caracter incluido en C[i], o de lo contrario, tendría (el cyber-delincuente) que llevar a cabo severas pruebas (exhaustivas) para conseguir el descifrado de datos [20]. Sin embargo, en Hexadecimal Random Caesar, no se adopta la incorporación de ruido en el empaquetado cifrado, solo se incluyen los componentes: C[i] y K[i] (seleccionado aleatorio con reemplazo, para aplicar desplazamientos sobre S[i] y generar C[i]), pero en esta nueva aportación, en lugar de realizar empaquetados del tipo ordinal/entero o caracter (ASCII), ahora se trabaja con su correspondiente número hexadecimal, empleando cuatro cifras, para permitir el cifrado de caracteres fuera del rango mod 255. El proceso de cifrado para Hexadecimal Random Caesar se puede definir formalmente como: PAQ_HRC = ((String) E) (mod 120) ; y para obtener el descifrado: D[i] = ((char)(PAQ[x] - PAQ[x+1])) (mod 120). Donde: E = EmpaquetadoFinal = (Hex2 ((int) C[i]) + Hex2 (K[i])) ; teniendo en cuenta que, en este caso, el signo "+" refiere a la función concatenar y que se trata de un par de valores en hexadecimal de cuatro cifras. La secuencia C[i] corresponde a la operación: C[i] = S[i] + K[i] (mod 120) ; mientras que S[i] es el texto plano y K[i] es el vector (aleatorio con reemplazo) que contiene los desplazamientos, uno para cada S[i] y cada valor K[i] debe ser menor que 105 (para no exceder el ASCII=255, ya que, en mod 120, el máximo valor es ASCII=150, por lo tanto, la operación: 105+150=255). Las funciones o "cast" String, int, char y Hex2 ; permiten convetir de hexadecimal a cadena de caracteres, de caracter ASCII a entero, de entero a caracter y entero a hexadecimal (de cuatro cifras), respectivamente. PAQ_HRC es un vector que guarda pares hexadecimales (de cuatro cifras) y contiene el paquete cifrado (incluye cada C[i] con su respectivo K[i]). El vector PAQ es generado por la asignación: PAQ = ((int) (SPLIT (PAQ_HRC , 4)). La función SPLIT divide a PAQ_HRC, convirtiéndola a vector de pares, en formato hexadecimal de cuatro cifras; mientras que PAQ[x] corresponde al valor de caracter cifrado C[i] y PAQ[x+1] es el correspondiente desplazamiento (K[i]).

El quinto método de cifrado utilizado y referido en esta investigación como: Noised Random Hexadecimal (Cifrado Aleatorio Con Ruido Hexadecimal), esta basado su procedimiento en la fusión de sus predecesores: Hexadecimal Random Caesar y Cifrado del Caesar por sustitución. Lo anterior significa, que se utiliza el paradigma basado en alfabetos, pero ahora, con formato hexadecimal. A diferencia de Random Caesar [20], el cifrado Hexadecimal Random Caesar no incluye la incorporación de "ruido intencional" en el empaquetado, solamente son concatenados (por pares hexadecimales de cuatro cifras) los valores correspondientes a la secuencia de caracteres cifrados C[i] acompañados cada uno de su respectivo desplazamiento K[i]. El único "ruido" que podría llegar a agregarse, sería de manera no intencional (aleatoria), por los caracteres no imprimibles que estuvieran fuera del rango mod 95, ya que, debemos recordar que Hexadecimal Random Caesar, trabaja con módulo 120 (es decir, los caracteres no deseables o con "ruido" serían en este caso, los que se encuentren dentro del rango: 30 a 32 y 127 a 150). En lo que concierne con Noised Random Hexadecimal, también trabaja con mod 120 y mediante el uso de números hexadecimales de cuatro cifras, pero debido a que, es un algoritmo de sustitución, ya no se emplea el vector de desplazamientos K[i], sino que, en su lugar, se definen dos alfabetos, uno para realizar el cifrado y otro para el descifrado. No debemos confundir el método Hexadecimal Random Caesar con la propuesta Noised Random Hexadecimal, ya que, difieren en el tamaño del empaquetado de cifrado, así como, en el proceso de encriptado y desencripción de datos. Podemos definir formalmente el proceso de cifrado con Noised Random Hexadecimal del siguiente modo: EMPAQUETADO = (String) (Hex2 (Alfabeto1[i]) + Hex2 (Alfabeto2[i]) + Hex2 (CifradoParcial[i])) (mod 120). Donde: El operador "+" refiere, en este caso, a la función concatenar de manera intercalada, de acuerdo con el contador "i", el cual, puede tener un valor máximo de 120, en caso de que la longitud del texto plano sea menor o igual que la longitud del Alfabeto1[i], de lo contrario, se actualiza "i" con el valor correspondiente a la longitud del texto plano a cifrar, que en esta investigación se utiliza como parámetro k (que no corresponde a desplazamientos). Los vectores: Alfabeto1[i] y Alfabeto2[i] contienen caracteres ASCII dentro del rango correspondiente a mod 120 (y por ende, su inicial longitud máxima es de 120 localidades, excepto si longitud de texto plano es mayor). El contenido de cada vector es seleccionado de manera aleatoria sin reemplazo (no se realizan comparaciones sobre el grado de similitud de los alfabetos, ya que, se entiende que en la práctica, es muy difícil que ambos alfabetos contengan la información en el mismo orden, salvo casos de excepción). Las funciones o "cast": Hex2 y String, permiten convertir un vector de ordinales o caracteres ASCII a formato hexadecimal de cuatro cifras; y, convierte un vector a cadena de caracteres, respectivamente. Por último, CifradoParcial[i] = Alfabeto2[i] , si y solo si, el texto plano S[k] se encuentra en Alfabeto1[i], de lo contrario se asigna: CifradoParcial[i] = Hex2 (S[k]). Cuando S[k] es longitud menor que Alfabeto1 [i], este último tiene longitud inicial de 120, por lo tanto, se tendrá que rellenar con "ruido" las localidades de CifradoParcial[i] (se trata de números aleatorios,

seleccionados con reemplazo, con formato hexadecimal); debe rellenarse, iniciando desde posición k hasta llegar al valor de "i". Si la longitud de S[k] es mayor que Alfabeto1 [i], en este caso, se tendrá que asignar i=k , insertando nuevas localidades a los vectores: Alfabeto1 [i] y Alfabeto2[i] , para rellenar con "ruido" (hexadecimal, aleatoriamente con reemplazo) estas nuevas localidades vacías. Por otra parte, para el descifrado de datos con Noised Random Hexadecimal, se utiliza la información del EMPAQUETADO, separando los vectores (que están intercalados) y buscando cada caracter del cifrado parcial en el Alfabeto2[i], regresando a su valor original localizado en la misma posición del Alfabeto1[i], siendo convertido de número hexadecimal a su valor de caracter, correspondiente en la tabla ASCII.

El sexto método de cifrado utilizado en esta investigación, denominado como: Noised Random 1-NN Hexadecimal (Cifrado Aleatorio Con Ruido 1-NN Hexadecimal), es considerado una nueva propuesta, debido a que no se reporta en la literatura su aplicación, o al menos no, con el mismo propósito que aquí se presenta. El cifrado Noised Random 1-NN Hexadecimal, es sucesor del método Noised Random Hexadecimal, y aunque el proceso para cifrado se lleva a cabo de manera similar, ambos modelos, difieren en el tamaño del empaquetado cifrado, debido a que, Noised Random 1-NN Hexadecimal, agrega un elemento adicional denominado Patrón[i], el cual, consiste en inyección de "ruido" (hexadecimal), siendo incorporado dentro del mismo empaquetado. Para realizar el cifrado con Noised Random 1-NN Hexadecimal, puede definirse formalmente como: Empaquetado1NNHex = (String) (Hex2 (Alfabeto1[i]) + Hex2 (Alfabeto2[i]) + Hex2 (CifradoParcial[i]) + Hex2 (Patrón [i])) (mod 120). Donde: El Empaquetado1NNHex es el paquete cifrado en formato hexadecimal de cuatro cifras (dicha conversión se realiza con la función o "cast": Hex2). Los primeros tres vectores del empaquetado (Alfabeto1[i] , Alfabeto2[i] y CifradoParcial[i]), son obtenidos de la misma forma que se lleva a cabo con Noised Random Hexadecimal; mientras que el Patron[i] es el vecino más cercano [13], [20], [22], [41]-[43] del CifradoParcial[i]. Suponiendo que ME[f][i+1] es una muestra de entrenamiento de (i + 1) columnas y f=100 filas (ya que "i" depende del tamaño de la longitud del texto plano S[k]). La posición (i + 1) refiere a la columna que guardará la información de etiqueta de clase (por usar una selección aleatoria con reemplazo, se entiende habrá oportunidad de repetir su valor, teniendo de este modo, una muestra de entrenamiento de varias clases). Cada fila es considerada como un patrón "ruidoso" que contiene ordinales o caracteres ASCII (seleccionados aleatoriamente con reemplazo con mod 120). Al buscar el vecino más cercano de CifradoParcial[i] en ME[f][i+1] se obtendrá como resultado Patrón[i] (es decir, aplicaremos la modalidad de eliminar su etiqueta para incorporar al empaquetado cifrado, ya que, la etiqueta solo sirve "para simular" que existe una distribución por clase). Este Patrón[i], es la secuencia de caracteres ordinales o ASCII más parecida al texto cifrado, y por ende, al ser incorporada (como "ruido") en el Empaquetado1NNHex cifrado, se entiende, que ello puede confundir a un "ciber-delincuente" que quisiera descifrar el mensaje. Por último, para el descifrado de datos, se separa la secuencia del Empaquetado1NNHex, buscando cada caracter del cifrado parcial en el Alfabeto2[i], siendo sustituido por el correspondiente en Alfabeto1[i], convirtiéndose el hexadecimal a valor caracter de la tabla ASCII.

Un último método utilizado, no es para realizar cifrado de datos, sino para evaluar los tiempos de ejecución y la estimación del error. El método utilizado se le conoce como: Método π (PI) o Validación Cruzada [9]-[13], [20], que consiste en los siguientes pasos: (1) Se extrae de la muestra de entrenamiento, un grupo de patrones 'ME-Pi' de tamaño 'Pi'. (2) El MODELO se entrena con la muestra de entrenamiento ('ME') sin incluir a 'ME-Pi'. (3) El modelo se entrena y/o evalúa con 'ME-Pi'. (4) El proceso se repite para: i = 1, 2, ..., (n / p). Donde: 'ME-Pi' se le conoce como muestra de control ('MC'), que sirve como prueba o test (en esta investigación, con la modalidad del encriptado de datos, no se realiza operaciones sobre la muestra de entrenamiento con este conjunto de datos 'MC', solo se extrae para poder realizar la estimación de la velocidad de cada modelo de cifrado, usando sesgo optimista); mientras que 'Pi' es el porcentaje extraído de la muestra de entrenamiento (en los experimentos fue utilizado un valor de 'Pi'=20%). El porcentaje para 'ME' utilizado fue del 80%. El modelo se refiere a cada método de cifrado de datos, siendo evaluado de manera separada. El término "entrenar o evaluar el modelo", en esta investigación, no se hace empleando 'ME-Pi', sino que se refiere a aplicar descifrado de datos sobre cada fila de la 'ME' (solo al 80% sin los patrones 'ME-Pi'), con el propósito de medir los tiempos de cifrado/descifrado, así como, observar si hubo algún error en el descifrado. Se entiende que 'p' es el porcentaje utilizado para 'ME-Pi', que va en función de n (tamaño de la ME). En esta investigación, el punto (4) se realiza repitiendo cinco veces la operación, siendo extraído de 'ME" un 20% de 'ME-Pi' distinto en cada repetición.

Adicionalmente, *con el propósito de comparar resultados, y poder validar las técnicas aquí propuestas, y su efectividad, respecto a otros métodos más tradicionales, fueron realizados nuevos experimentos usando las mismas muestras de datos, referidas previamente, haciendo uso de la validación cruzada, descrita anticipadamente, trabajando de manera separada, cada uno de los siguientes métodos o algoritmos de cifrado de datos:* AES

(Advanced Encryption Standard [25], [36], [39]), RSA (Rivest-Shamir-Adleman [2],[25],[36],[39]), 3DES (Triple Data Encryption Standard [2]), RC (Rivest Cipher [2]) y DES (Data Encryption Standard [2],[25],[36],[39]). *La versión empleada de AES [2],[36] fue: AES-256 [44]-[45]. La variante utilizada para RC [2],[36] ha sido la denominada como: RC4 (Rivest Cipher 4) [44]-[45], y con respecto a RSA [2],[25],[36],[39], se ha utilizado: RSA-2048 [44]-[45].*

3 Resultados y discusión

Los experimentos realizados con los métodos de cifrado, descritos previamente, fueron llevados a cabo utilizando muestras de entrenamiento (ME) generadas de manera aleatoria con reemplazo, donde cada fila contiene el texto cifrado, cuya etiqueta de clase corresponde al mismo mensaje de texto original (sin descifrar); tal como se ha descrito previamente en la sección de la metodología. Cabe agregar, que el número máximo de filas en la ME fue de 1000 ejemplares y el tamaño máximo utilizado como columnas corresponde a la longitud de la cadenas de texto a cifrar, sin rebasar el límite de 255 caracteres (excepto el algoritmo 3DES, que usa estrictamente una longitud igual a 24 bits), se agregó dos columnas adicionales: una para conocer el tiempo de cifrado (en milisegundos) y otra columna para poder indicar posteriormente el error, y de esta forma, facilitar el cálculo de los promedios. Del mismo modo, cada cadena de texto a cifrar, puede ser almacenada como vector o texto plano (aunque en este caso, fue necesario convertir a vector tipo ordinal o entero o caracteres tipo ASCII o hexadecimal de cuatro cifras, según el método de cifrado empleado). Cabe aclarar, que para cada método de cifrado utilizado, se diseñó su propia ME y fueron evaluados cada uno de ellos, de manera separada. El procedimiento empleado para medir la estimación del error, así como, para estimar la velocidad promedio de cada método de cifrado de datos, fue la Validación Cruzada [9]-[13], [20], siendo aplicado de la forma en que se describe en la metodología.

Durante el inicio de la experimentación, fueron estudiados los métodos de cifrado que no cuentan con desplazamiento K[i] aleatorio, los cuales, se aplicaron a la muestra de entrenamiento correspondiente, para cada uno de ellos; tal es el caso del clásico Cifrado Caesar Por Desplazamiento De K, el tradicional Cifrado Del César Por Sustitución (usando módulo 26, en ambos casos) y Hexadecimal Caesar (usando módulo 129). Los valores de desplazamiento K utilizados para la aplicación de estos métodos sobre su muestra de entrenamiento particular fueron: K=4, K=11, K=95, K=120 y K=255; lo anterior, para poder comparar resultados con otras investigaciones [20], [36]-[37], esta información la podemos apreciar en Tabla 1. Posteriormente, se estudiaron y llevaron a cabo los experimentos, con los métodos de cifrado, cuyo valor de desplazamiento K[i] es aleatorio con reemplazo, dichos modelos son: Hexadecimal Random Caesar, Noised Random Hexadecimal y Noised Random 1-NN Hexadecimal, todos ellos usando módulo 120; dichos resultados se muestran en la Tabla 2. Los experimentos descritos anteriormente, fueron realizados con el programa de cómputo escrito en lenguaje Java; y se repitieron utilizando el programa de cómputo en lenguaje Python 3, observando que no hubo diferencia significativa en los resultados. En su mayoría, los experimentos tuvieron éxito; sin embargo, se considera, que falta profundizar un poco más, extendiendo el uso de cadenas de texto plano con longitud mayor que 255 caracteres. Por último, *con el propósito de comparar resultados, y poder validar las técnicas aquí propuestas, y su efectividad, respecto a otros métodos más tradicionales, fueron realizados nuevos experimentos usando las mismas muestras de datos, referidas previamente, haciendo uso de la validación cruzada, descrita anticipadamente, trabajando de manera separada, de los siguientes métodos o algoritmos de cifrado de datos: AES (Advanced Encryption Standard [25], [36], [39]), RSA (Rivest-Shamir-Adleman [2],[25],[36],[39]), 3DES (Triple Data Encryption Standard [2]), RC (Rivest Cipher [2]) y DES (Data Encryption Standard [2],[25],[36],[39]). Solo que en este caso, los experimentos fueron realizados, implementando, solamente en lenguaje Python 3, cada uno de los algoritmos, antes mencionados (ver Tabla 3). En los experimentos realizados con AES-256, los parámetros asignados fueron los siguientes: uso del modo de encadenamiento de bloque de cifrado, conocido como: CBC (Cipher Block Chaining Mode), con un vector de inicialización de IV= "0000000000000001" y uso del estándar de criptografía de llave pública: PKCS7 (Public Key Cryptography Standard #7), como método de relleno (padding) con 128 bits, empleando codificación en formato hexadecimal, un tamaño de clave (KeySize) de 256 bits y una clave secreta (Key) con valor de "0000000000000000000000000000001". Para el caso del RSA, durante la experimentación se utilizó formato de archivo de clave criptográfica de correo privado mejorado, conocido como: PEM (Privacy-Enhanced Mail), con extensión de 2048 para definición de clave pública y privada (Public/Private Key RSA 2048), haciendo uso del estándar de criptografía de llave pública basado en relleno (padding) de cifrado asimétrico óptimo, conocido como: PKCS1_OAEP (Public Key Cryptography Standard with Optimal Asymmetric Encryption Padding), utilizando una*

clave secreta (Key) con valor de "00000000000000000000000000000001", obteniendo como resultado cifrado, una secuencia en formato hexadecimal.

Tabla 1: Resultados preliminares de los métodos de cifrado estudiados que cuentan con desplazamiento de K estático (ejemplo usando como texto plano: we will meet at Midnight)

ALGORITMO DE CIFRADO	VALOR DE K	TIEMPO (ms)	ERROR (%)	TEXTO CIFRADO
Tradicional Cifrado Caesar Por Desplazamiento De K	4	59	1	AI AMPP QIIX EX QMHRMKLX
	11	30	1	HP HTWW XPPE LE XTOYTRSE
	95	47	2	WE WILL MEET AT MIDNIGHT
	120	31	2	WE WILL MEET AT MIDNIGHT
	255	29	2	WE WILL MEET AT MIDNIGHT
	Promedio	**39.2**	**1.6**	
Tradicional Cifrado Del Cesar Por Sustitución	4	17	0	ai ampp qiix ex Qmhrmklx
	11	10	0	hp htww xppe le Xtoytrse
	95	10	1	we will meet at Midnight
	120	10	1	we will meet at Midnight
	255	10	1	we will meet at Midnight
	Promedio	**11.4**	**0.6**	
Cifrado Caesar Hexadecimal	4	4	0	007b00690024007b006d00700070259700240071006900690078002400650078002400510 06d00680072006d006b006c00782597
	11	4	0	00820070002b00820074007700772 59e002b00780070007000 7f002b00580074006f007900740072007 3007f259e
	95	3	0	00d600c4007f00d600c800cb00cb25f2007f00cc00c400c400d3007f00c000d3007f00ac00c800c300cd00c800c600c700d325f2
	120	4	0	00ef00dd009800ef00e100e400e4260b009800e500dd00dd00ec009800d900ec009800c500e100dc00e600e100df00e000ec260b
	255	4	0	00f800e600a100f800ea00ed00ed261400a100ee00e600e600f500a100e200f500a100ce00ea00e500ef00ea00e800e900f52614
	Promedio	**3.8**	**0**	
TOTAL :		**18.1333**	**0.7333**	

En los experimentos con RSA-2048, se utilizó como clave pública la siguiente: "-----BEGIN PUBLIC KEY-----.Proc-Type: 4,ENCRYPTED.DEK-Info:DES-EDE3-CBC,17BCFA414000D9A4xHqpW1evXi+q0MRPBfeZS9vQaGz/416cNrMIxMzBJ53G2R8psQza2bFOxds/MOLYqt3IwlV04Bvihv7lVqoLDfPD0287 28oULYSehkCjsmUGgRl7v+Pe4cfz/ODhIGdNuS98JsTDZxUGbtoSvJoP/vS7+BISCwcLrrpt6BKtkkzu/1j2LG+W5J+7WqE3fnEtW9dn2y0Wbx044Th/ RzLhUHhzmfWpEB5oRSF94WAzX9l01M2O954RWxR2sZUxJ+iF/46ruiz4L5EbDI5j9PUI7VxqJXwDemWDa8SJ9+mLY5Vz1NNh8Yl2Gh++PEu2J DUPNQB2CRAx/j03+2nUuumHy6/XA/LkPeWGw9ULzaOTHXM1a1kExbLhTy//kCFqQCd+6wSZLQVt2Hg= -----END PUBLIC KEY-----". *La clave privada que se ha empleado, se muestra enseguida: "* -----BEGIN RSA PRIVATE KEY----- Proc-Type: 4, ENCRYPTED DEK-Info: DES-EDE3-CBC, 2C3EEE4347B3AA0C5QnfB6QTy2PttNbDcMg5f+i7sfSuUw+WRwezScOhQIsi9BtlMYmitEeMjxpT/eH/DzNnBCPguByVPvwAcJT1eZrehLBrqy4Fk by5jd5CVDXMVpYjNdRUoVy4TDVb2Y4HEMHrxpkqE08BwHNLQM5ULl/0tIJglDeyJbsOa2/mIFpMB2OEMFO9wDZka3hQPZesmFlt+EJmpEUD ni0UnyvUqzYC4zLOmWKHfcZ8NbPl3lyW+DyvWGq3CIjE7Flx+nyDB9MXylKyaBhfOQiuAcaQ3Yj4bYKvczxTBf/hSX7n9DdgejXhTnEPUaW8eX RAgXBdVgvf3oyLe8ygktS2o75aR4zcDmIjflm/eCLrQPB8qv6arDp7lOvsLdCzToAnU1JciLcEi4LV5gajDoPJYfaVrvF+ZYsBdybKpf6hg31txdAZxA MErDFXwp/+B6021bk85x+MW440LY7BjYITPE+EXt9aCb7AiKRaredKyjM+/2bxkkOQ4HF8fkLNvQrKeggLF1xQUlMUu9I9qBjQR1pztBmhyxC2 Hs+HnVFFvM+mtoIgDOZtj7NrOi+vNxnjUr4+q405R7MKWSRUlA5fQFf7UHDbC2FtpLxrWPXc4UvHYpYK9caMbwjqJqzUlgUo0FZz6bopv0eCkT GccyItOJ5VZAI8BkxseAAC3UNeG85tdhBQ3/zbBqiM98dR3GNzVor6CaDqqBvbhDUBe+ef/k6ScxVjtx/xUu+VBEGD1sMTB+bjlwRmTifhnqyXeG2 mx8RGQrUhWMOFrDmegxl1WQ+zKD0aJ2QzL56VyGP01DuJtx8QfijRKvbtVzD8qQgCpv5Mz2WBCYhfrxT4DdDxDSER4FKoyK+Tx65WGiJWK MrA1rR+mR1nGnoziVa7XRWAzT7swTXkwtKkYOcrv3fhEHPmsdy/BpMDFjw+JvTD3tdDOpRj5zG2yZFvK9WJTAB+fWGaG5HY4QgG2i4UrokG lyHC9afXTWUa4Nnj6nYq7ksVppGccnsP3ZI+8kL6Mh4ho+538xr4ZY0RPUxqyHUrHQN6sjrCg/3L/ahgmh9oclIC8gQozf6EY1AqhX7nCiWevTmFJh makEWZq4LB3DyyYrPCOjJA8dDVaCbERhEw2DydM6bp/MHjor0X8RkeygxfhcCc6p8KMxfADu/92cWaw1GAkpFh1iBmmj6uaVZZy+bnAN1jA8t 0srcv4E6Ff0ZmsZD0QdXazjXplG69aIMK3Snd8nT/dE86GqMgKoCYNAXeF+T1fa+AiEV4JuJIuRrZjknmoBFl+4qIZzUgbH6MsxKtZ5SjZgzjaZdEX YpZwecnzPIOm+QdQtfpQbUJMsMdT4iwBtzJfKSqvCD05Ax3sifTfLzdyWvOvip9eEXFr9hTLSzgC88a922oqycp3erXQ/OGTFDmTrsIohu9U1hGTup T4mciErCYKoU/hmvHLjRbZRXan+1oVcHuE4xrrTYRz9tB+JzxjQ1YJsPGo8nF3f+qV9/jV6WjzZPKd47QE4sBz++VPRZWEVso/Ek0pkljSXzACCly 1OxrEu9oWr3UCm5nIoXym+ttPxbXjJYLLaZ/vARB5lQb3QCiydihsDxWfLCw -----END RSA PRIVATE KEY-----*". Estas claves fueron generadas en lenguaje Python 3, utilizando la librería "pycryptodome" [46]. En lo que concierne con la experimentación de RC4, se utilizaron valores estándar definidos por la librería "pycryptodome" [46] de Python 3. También, fue empleado el formato hexadecimal con codificación: UTF-8 y como clave secreta (de 128 bits), se asignó un valor de "00000001".

Tabla 2: Resultados preliminares de los métodos de cifrado estudiados que cuentan con desplazamiento de K dinámico (ejemplo usando como texto plano: we will▒ meet at Midnight▒)

ALGORITMO DE CIFRADO	Test	TIEMPO(ms)	ERROR (%)	TEXTO CIFRADO
Hexadecimal Random Caesar	Test 1	11	0	00e0006900c3005e004a002a00c9005200ac004300ae004200d5006925d70044005700 3700b4004700ce00690094002f00dd00690058003800a1004000960022005900 3900b6006900d2006900cd006900d70069009e003500b4004d00d1006900c6005225fc0069
	Test 2	10	0	00a2002b0093002e0052003200b900420091002800b3004700d5006925c5003200890069009002300b4004f00be005900dd00690089006900 8e002d00ca0056008900690090004300 8e002500b3004f00b40046009b003200c2005b00d1006900dd006925ba0027
	Promedio	**10.5**	**0**	
Noised Random Hexadecimal	Test 1	88	0	004d006f007a0078003e00640023008e006c0077007a007a0030002d003c0056008300 3a003a0060003a00740026259300520061006c0035005e0028006500640064005006e0064005a005300260043004800 6c0047004d0087003800890026003600270 06c0040007d006f00790065003c009200910035004500 7c0021007d008d003c0086004c003200490041002500 4c003f00260093009225930 08e008500fe0025006200fd0063007500fe0050003700fd008a008b00ff0021005100fd0027006b00fd0095004700fe0069003c00fd00240023006e0068002500ff003e004a00ff0039007700fd0057007200fe007b003000fe002e007100ff0032001f00fe0088005f00fe0046006900fe005f003400fe004b003d00fe0084005200fd006a008400ff00890036 00fd0083007e00fe0051006600fd0087008000ff005d006800fe008f008600ff0082004600fe0054008c00fe006e002100fd007f006d00ff003f007400fd006f002900fd002f004b00ff0080006300ff005e005700fd0075005a00ff0094006700fe008d004200ff0066004500fe0070002c00ff007a009300ff007e002e00fd0061008700ff0029001e00fe002b007900fe0020006c00fd0076005400ff0022004300fe0044005800fe0059009400fd002a005b00ff001f005600fd002d003900fe004f007300ff0037003300ff0048009500fd0072008100fd006b008a00fe002c004f00fe0060003800fe004a002a00fd0033005c00fd0053009000ff003d002f00fe0041007b00fe004e005d00ff003b003100ff006c003a00ff0031004900fd0090002000fe0071002400fd0026007800fe0073008200fd0085007f00fd0042007600fe005b004e00fd0034002200fe0028008f00fd005c004400fe0081002b00fe006d0028 00fe003c005500ff008c005000ff008b007000ff0091008800fd0062006a00fd007c004000ff0064003500fd001e005900ff0067003200ff0058003b00fe
	Test 2	96	0	008c0035004c0028009000590080004900870041004 1004c0058006a004d0090002600380068004500380023005d2593008f008c00870046008d002d00780068005 9001e007c0059005a005a001e007a00290087002c008e0046006b0025001e0025006200870026003900 4b0045005f004d0031003e005c003c0072003b0073003f004d0043002b003c003f00670045004a006d001e0022006025930035003100fd0075002a00fd0077004c00fd004e002700fe007e008900fd0049008000ff0094007b00ff0021007a00fd0055005100fd002f006f00fd0067003c00fe0034008400fe0083008200ff0072004f00fe006f002100fe0069004d00ff0091002200fe005e001f00fe0071008500fe0070006600fe0040004700fe0081005b00fe0095007500ff003d005e00fe0063006b00fe0037002800fe0027007900ff005d007300fd0089003200ff0024003a00ff0093004a00ff0057007600ff004d004b00ff003b002c00fe008b006400ff0088007400fe0048006c00fe003e006100fd0042002400ff008a006300fd0052005800fe008d007f00ff0050008100fe0053004000fd006c003800ff0036009100fd004f003700fe005f007e00ff0085008b00fd0020008700fd0065005900ff0059004200fd006a008800fd0064005c00ff0087007700fd008e005200fd0044008f00fe002b004e00fd002d002300fe005b003400fe0056009400fe0039005300fe007b004800fe0066002000fd0082009200ff0092005700fe007d006e00ff002e007d00fe0074001e00fd00860078 00fe002a003d00ff0054003600fe0032005600ff006e003b00fd0033002e00fd0107c008300fe005c009500fe0076004400fe004c009300ff003a005500ff006d002d00fd0051004300ff0029005000ff0079006900fd007f008a00ff0047002f00fe001f003000ff0061004600fd004b005400ff0084007100fe0038008600fd0060007000fd0062003300fd003006500ff
	Promedio	**92**	**0**	
Noised Random 1-NN Hexadecimal	Test 1	101	0	0029007b004d008c0080005d0028007f007b0090002400270 03b0042004d0022005b005a002b005f006b00750088008700340051008800 2a0075005925930094003500480024007700240 08b003d006100890069002800460046007700280068004300540 08e0041008700530024006b001f007d004b004a002f001f008e0026003800570024004d008b0087006a00270055003e003e002100 3808f0025004a00760043007e0070002b007f0052002600640 08c0056005c0022007f0077004d008e0086006300682593003e002a003100fe00fd0094004100fe00fe004f006b00fd00fd00600089 00fd00fd006f002e00fd00fd0074008e00fd00ff004b008000ff00ff008a009400fd00fe0042007f00ff00fe0090004500ff00fd00730023 00ff00ff0082004600fe00fe005e003300fd00fe0021008d00ff00ff0027007c00ff00fd005a003900ff00ff002d002700ff00fe004e007100fd00fe0083007e00fe00fd0086008600fe00fd003d007400fd00ff0032008400ff00fd0071001e00fd00ff008d003f00fe00fe0092002a00fe00fd0057005b00fd00fe0033006f00fe00ff0022004300fd00ff002b005500fe00ff0053006700fd00fd003a006200fd00fe004d006a00ff00fd0091004400fd00fe006d003d00fe00fd0030008c00fd00fe0040006500ff00ff0093003200fd00fd0064003800fd00fd003c003400fe00fd0037003b00fe00ff0059007800fd00fe004a005800ff00ff0061004b00ff00ff0041003500ff00fd0031002000fe00ff005f002900fe00fd0051006000ff00ff0065002800ff00ff0058003600fe00fd0045008300fe00ff0078008a00ff00ff006c00880ff00ff0067006400fd00ff0069002b00fe00fd0050003c00ff00fd007d009200ff00fd0095008500fd00fd0054007900fe00fe0079005f00ff00fe0026007300fe00ff0081006300fe00ff006a003a00fd00ff003f008200ff00ff0044002d00fd00fd002c009300fe00fd0048003700fd00fe0036004c00ff00fd0084003000fd00ff0066006c00ff00ff007c004e00ff00ff008f002f00fd00ff0062004900fe00fe007a002500fe00fe004c005600fd00fe008c007200fd00fe0023004f00fd00fd0047008100fe00fe0020002400fe00ff008e005200ff00ff0072006d00fd00fe0049002c00ff00fe005c007a00fd00fd0070006100fd00fd0088004700fe00fe006e007600fd00fe005d008f00fe00fe0039005e00fd00fd0068002200fd00fe0076005000fe00ff0028009500fe00fd002e009100fe00fe0085004000fe00fe007f006600fd00fd001e006e00fe00fd
	Test 2	96	0	006f007f005f0092005f004b0066004800500063003a0031003300920 05f0039006c006d0051007e004f0060006d00720027008200 6d0078003200212593009500540093003a004a0072007d005b008c0067001f00660081005c002d0066008b008600900095007c005b004f003a0035004500 8c008e0091002b003100950081004c004a003a005e001e0083005c008800390074005100440078002200 4d0091007d007a0085003800340061005100630058005600 1f0094006100 8e0076003e0094006500950034009200 8d259300290038006400fe00fe004d005c00ff00ff0028004400fe00fe0048007800fe00fe0075008600fe00fe0068007600fe00ff006a007700fd00ff0064004d00fe00ff0063005000ff00fd0041005800fe00fd0057003b00fe00fe0059007e00fd00fe0026002000ff00fd0021006e00fe00ff005e008b00fe00ff008c007100fd00fe0035004c00fd00ff008e001e00ff00fd0036006f00fd00fe0055006900fd00fe008f005e00fe00fd0065006600fd00fd0023003000ff00fe0049002e00fe00fe0084004500fd00fd003d002b00ff00fd007b005200fd00ff005a003600ff00fe002e004100ff00fe0043006800ff00ff007f006700ff00ff0073003200fe00ff0091007c00fd00ff0071003d00ff00fd0024007000fd00ff006e008500fd00ff0020003a00fe00ff003c004e00fe00fe002a008000ff00ff0089005900fe00fd001f004000fe00fe0031003300fd00fe0069005100fe00fe0029003e00fe00fe0080003800ff00fe0025004600fe00ff0042003f00fe00ff0074009500fe00fe007c002600fd00ff0077005f00fd00ff008a003400fd00fd0090004900fe00ff0079004300fd00fd0047006b00fd00fe0030009400fd00fe006d005b00ff00ff003e005300fe00fd003f007300fe00ff0081003500fd00fe0060004800fe00fe007e003900fe00fe0040005500ff00fd0037006200fd00fe0022008900fe00ff0062008400ff00fd0056007b00fd00ff0087009100fd00ff002f005400fe00fd0051002500ff00ff0077006c00ff00fd007a008a00fe00fd0038008800fd00ff0052006a00fe00fd0095007200ff00ff0044002300fe00fe0085004700fe00fd0076002400fe00ff003a002900ff00fd003b002700fe00ff004a004200fe00fe004b003700fe00fd006b002c00ff00fd008d005a00ff00ff004e002f00fd00ff008b002a00fd00ff0053005700fe00ff0066007900fe00ff002d007500fe00fd0093003c00fe00fe0082008100fd00fd002c005d00fe00fe005d008f00ff00ff0046002800fd00fe0088008700ff00ff
	Promedio	**98.5**	**0**	
TOTAL :		**67**	**0**	

Para el caso del algoritmo 3DES, se usó el modo de libro de código electrónico, conocido como: ECB (Electronic Codebook Mode), con valor de "000000000000000000000001" como clave (secreta) de 24 bits, obteniendo la secuencia cifrada en formato hexadecimal. Finalmente, los resultados cifrados para el algoritmo: DES, también

fueron en formato hexadecimal con codificación UTF-8, empleando una clave de 56bits con el siguiente valor: "00000001" y usando el modo de libro de código electrónico, conocido como: ECB (Electronic Codebook Mode). En lo que concierne a los tiempos de ejecución de cifrado/descifrado, fueron calculados para cada texto plano, en cada repetición de la validación cruzada, y al final, se hizo un promedio. Los errores, fueron considerados al momento de descifrado. Estamos en el entendido que la ME contiene el texto cifrado, acompañado del tiempo de cifrado y una etiqueta que corresponde al texto plano a cifrar (así como, la columna que guarda el error, pero ello se hace en tiempo de ejecución, ya que, inicialmente tiene cero). Por lo tanto, se procede descifrando el texto y si no es igual que la etiqueta de clase, se procede a anotar un error en la columna correspondiente de la ME, todo ello, es coordinado por cada repetición de la validación cruzada.

Tabla 3: Resultados preliminares usando algunos métodos o algoritmos de cifrado tradicionales (un ejemplo de muestra, usando como texto plano: we will▒ meet at Midnight▒)

ALGORITMO DE CIFRADO	Test	TIEMPO(ms)	ERROR (%)	TEXTO CIFRADO
AES-256	Test 1	57	0	30303030303030303030303030303030314e86f8918b532c730b96262c2a7739bade2d5c68c7e3e219bc5a943d0ebd0cc0
	Test 2	97	0	30303030303030303030303030303030314e86f8918b532c730b96262c2a7739bade2d5c68c7e3e219bc5a943d0ebd0cc0
	Promedio	**77.0**	**0**	
RSA-2048	Test 1	96	0	8ef526e608728054fa749463334a6e6bc7cb853aac33ea9e6612e2ddb9ccf7f2172ec8e87c3b893fd83dd0db6af04ab5118e25434280b7bcc4d03f34f029334470d09532a5663417bb092772e1ba44552f53d7f05059cc1cdbcea5dc917b0762f4095974b222d36c6dd73e7b8313d55649c746b37238c94c809b00caed28590156c81329ebeff90986167ae916ee2a5596c1ef95afe47f57a5df8c947922418dbc67df6dfa1d700b1d0cceeddddeca9e0cdf3f72f8fd2ac97c2b4c33c79de4c063251069957d4d7ed58e0988838773d990368671947385925 9be20aaf0aa09d6f0a1eba818beceb426393b204a6c42cc7a4d440f88aee97880ffaed78e93e8991
	Test 2	121	0	b4de0256e0f25f994d03f30a130887f7074fc7a181dbb11c2d0d5d5d9b0234686ed546318c10bb0b1100fe05881d59104bcf838e3817dfbfe9a303d2ac3eeaef37473b3f3b6c8461038a05d2408ff22f3d5c6498b9fe77740794aab59b0ff04174400719634c384ada8b19d619ed804c8f58bacfac1eec3897bb9057fc80cf27c108d8ab07d91fc828c0abedbd0556987a1789c9df6d963c23d6f5c8c411e75ce18bc129b1022c9899ff24c64493c0d5882a2b0e6fde129a84908b4f0226a476ebbeb7fbb949978d5ccae4b5cd3ea163ee8f728a17b6cb32070f0998af86c406824de74fe9e85f2c379b5e94238375cfdf05a614e1b16aa1eea05e295e934c64
	Promedio	**108.5**	**0**	
3DES	Test 1	55	1	3e00ddc198431704c76f0db605a0a7cacfb3dc742edbeba7
	Test 2	122	0	3e00ddc198431704c76f0db605a0a7cacfb3dc742edbeba7
	Promedio	**88.5**	**50**	
RC4	Test 1	20	0	949f2244e3ab422b0d24ea69bed4da930d0a89b2fdf9944895b73f9182666fd2
	Test 2	31	0	949f2244e3ab422b0d24ea69bed4da930d0a89b2fdf9944895b73f9182666fd2
	Promedio	**25.5**	**0**	
DES	Test 1	28	0	3e00ddc198431704c76f0db605a0a7cacfb3dc742edbeba7f0a688c1d906a355
	Test 2	39	0	3e00ddc198431704c76f0db605a0a7cacfb3dc742edbeba7f0a688c1d906a355
	Promedio	**33.5**	**0**	
TOTAL :		**66.6**	**10**	

4 Conclusión

Los métodos de cifrado de datos con desplazamiento K estático, aquí estudiados (ver Tabla 1), aunque en algunos casos, su proceso de encriptado es más rápido que las propuestas basadas en aleatoriedad (cifrado dinámico), así como, los algoritmos tradicionales, se puede apreciar vulnerable la seguridad de dichos métodos estáticos, debido a que, no presentan variaciones en el contenido para una misma cadena de texto plano a cifrar. En otras investigaciones [20] se señala que este tipo de casos podrían ser descifrados usando un diccionario con uso de procesamiento de lenguaje natural mediante inteligencia artificial. Además, dichas propuestas, cuando se usa módulos 95, 120 y 255, no se logra cifrar el texto plano (ver Tabla 1). En cambio, los métodos de cifrado basados en aleatoriedad, ya sea, por sustitución o por desplazamiento de K[i], se observa (ver Tabla 2), que en algunos casos, la diferencia promedio de velocidades para el cifrado de datos, es mayor, pero no sustancial, ya que, no tarda en ningún caso más de un segundo (1000 ms). Sin embargo, hay que recordar que los textos planos empleados son menores de 255 caracteres. Empero, resulta más segura la información con cifrado aleatorio, ya que, el contenido del paquete, para una misma cadena de texto, produce distinto resultado en cada ejecución, mientras que en las propuestas con inyección de ruido hexadecimal, genera un paquete cifrado de mayor tamaño, pero ello lo hace menos vulnerable a ataques, en

comparación con el resto de los métodos, evaluados en esta investigación. Una desventaja de los métodos de cifrado basados en aleatoriedad, aquí estudiados, es que pueden llegar a seleccionar, carácteres de la tabla ASCII que son no imprimibles en pantalla, ello puede producir pérdida de información. *Sin embargo, en esta investigación, las propuestas basadas en módulo 95 y módulo 120, no presentaron dicho problema, al ser utilizados números hexadecimales de cuatro cifras. En cambio, las técnicas o algoritmos tradicionales para el cifrado de datos, los aquí estudiados, solamente uno de ellos (RSA-2048), logró reportar cifrados dinámicos (ver Tabla 3), pero demora más tiempo en el proceso de cifrado, en comparación con las propuestas aleatorias, basadas en hexadecimal, aquí estudiadas. Los resultados obtenidos con AES-256, no muestran ser dinámicos, ya que, genera siempre la misma secuencia de cifrado como resultado, cuando se utilizan los mismos valores de la clave secreta y vector de inicialización. Sin embargo, ello no indica que el cifrado sea inseguro, ya que, en los experimentos se observó un 0% de error. También, el algoritmo 3DES, no presenta resultados dinámicos en la secuencia cifrada. Además, el proceso de cifrado tardó más tiempo que el reportado por AES-256. Adicionalmente, se observó la presencia de un 50% de error en el descrifrado de datos, cuando el texto plano de entrada contiene caracteres, cuyo valor ordinal está fuera del rango de la tabla ASCII (en el ejemplo de prueba de la Tabla 3, se presenta en el texto de entrada el caracter: ▒'). Otro factor observado, fue la presencia de error, cuando se utilizan entradas de texto plano mayores al definido por 3DES, que en esta investigación se han delimitado a 24 bits (en este caso, se obtiene 100% de error, al ser cortadas las secuencias de texto de entrada y ajustada a 24 bits). Del mismo modo, el algoritmo RC4, aunque no reporta porcentajes de error, se observó que los resultados cifrados no son dinámicos. A pesar que su proceso de cifrado es mucho más rápido (más del 50%) comparado con: AES-256, 3DES y RSA-2048. Adicionalmente, los resultados de la secuencia cifrada reportados por el algoritmo DES, no presentan características dinámicas, aunque se obtuvo un 0% de error. A pesar que su proceso de cifrado es más rápido que los algoritmos: RSA-2048, AES-256 y 3DES, no logró superar en velocidad al RC4. En cambio, utilizando RSA-2048, se observó que regresa un resultado dinámico, aunque tarda más tiempo en el proceso de cifrado que los algoritmos: AES-256, RC4, DES y 3DES. En este caso, también se observó un 0% de error. Por último, con respecto a las nuevas propuestas dinámicas, basadas en formato hexadecimal, con inyección de ruido, y teniendo en cuenta que los algoritmos: AES-256, 3DES, RC4 y DES, no reportan resultados de cifrados dinámicos, en este contexto, los denominados como: "Noised Random", superan las expectativas, en esta investigación, porque reportan siempre resultados dinámicos distintos en cada ejecución del algoritmo, y aunque tardan más tiempo que: AES-256, 3DES, RC4 y DES, en el proceso de cifrado de datos, superan la velocidad de los tiempos reportados por el algoritmo: RSA-2048 (ver Tabla 2 y Tabla 3), que en esta investigación, fue el único algoritmo de los métodos tradicionales, aquí estudiados, que logró reportar secuencias de cifrado dinámicas distintas en cada experimento, modo similar a lo observado con los métodos: "Noised Random" y "Hexadecimal Random Caesar". Sin embargo, esta última propuesta "no ruidosa", además de reportar secuencias de cifrado más nítidas (menos extensas), logró superar en velocidad, al resto de las técnicas (de cifrado dinámicas, aleatorias y tradicionales) en los tiempos para el encriptado de datos, entre un 2.4% y 10.33 % veces más rápido.*

En general, todos los métodos de cifrado, aquí estudiados, reportan buen margen de acierto (excepto los algoritmos: 3DES y Cifrado Caesar con desplazamiento por K); y, la velocidad de cifrado, en promedio difieren aproximadamente entre 35.4% y 88% milisegundos, para el caso de los métodos basados en desplazamiento de K y K[i] aleatorio. Por último, observando resultados presentados por otros autores [20], podemos aprecia que discute acerca del uso de un nuevo formato denominado Pseudo-Hexadecimal, el cual, se introduce a los alfabetos en el paquete de cifrado, con la intención de inyectar ruido, este podría ser un trabajo futuro a desarrollar, llevando a cabo la aplicación de los métodos aquí estudiados, usando dicho formato Pseudo-Hexadecimal.

Agradecimientos

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

Referencias

[1] 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.

[2] Mendoza, J.C. (2008). Demostración De Cifrado Simetrico Y Asimetrico. Ingenius. Revista de Ciencia y Tecnología, núm. 3, pp. 46-53. Universidad Politécnica Salesian. Cuenca, Ecuador. ISSN: 1390-650X. Disponible en: http://www.redalyc.org/articulo.oa?id=505554806007.

[3] Kalsi, S., Kaur, H., & 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

[4] 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.

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

[6] Singh, S. (2000). Los códigos secretos. Madrid: Debate.

[7] 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. Disponible en: http://www.redalyc.org/articulo.oa?id=533662765008

[8] Hebert, S. (s.f.). A Brief History of Cryptography. Disponible en: http://cybercrimes.net/aindex.html

[9] Rangel, E. (2002). Vecinos Envolventes para Variantes de la Regla del Vecino más Cercano. MS Thesis, Instituto Tecnológico de Toluca, México. ["Variants For Nearest Centroid Neighbour"].

[10] Rangel, E., & Barandela, R. (2004). Nearest Centroid Neighbour, An Alternative in Pattern Recognition for Detecting New Tasks in a Mobile Robot Simulator. Enviado a: 11th International Congress On Computer Science Research (CIICC04). September 31, October 1, 2. Ciudad de México - México (Artículo En Extenso). Disponible en: http://erangel.coolpage.biz/pappers/p2004.jpg

[11] Rangel, E., & Rodríguez, C. (2018). Un Estudio Con Variantes De La Regla NN, Como Alternativa En Inteligencia Artificial Para Incrementar La Precisión En Clasificación De Patrones. Publicado En: Primer Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Tecnológico Nacional De México (campus: Instituto Tecnológico de Cd. Altamirano). Noviembre, 2018. Cd. Altamirano, Estado De Guerrero, México. (Artículo En Extenso). Disponible en: http://erangel.coolpage.biz/pappers/p2018b.jpg

[12] Rangel, E. (2019). Resultados Preliminares Con Variantes De La Regla NN, Como Alternativa En Inteligencia Artificial, Para Clasificación Usando Muestras De Entrenamiento Desbalanceadas. Publicado En: Segundo Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Tecnológico Nacional De México (campus: Instituto Tecnológico de Cd. Altamirano). Noviembre, 2019. Cd. Altamirano, Estado De Guerrero, México. (Artículo En Extenso). Disponible en: http://erangel.coolpage.biz/pappers/p2019.jpg

[13] 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. Publicado 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). Disponible en: http://erangel.coolpage.biz/pappers/edgarrangel2022.pdf

[14] 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.

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

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

[17] 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.

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

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

[20] 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. Tecnológico Nacional De México, Instituto Tecnológico de Cd. Altamirano. Sexto (VI) Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Noviembre, 2023. Cd. Altamirano, Estado De Guerrero, México. Disponible en: https://www.cdaltamirano.tecnm.mx/index.php/17-vi-congreso-nacional-de-investigacion-en-ciencia-e-innovacion-de-tecnologias-productivas/140-tecnm-40

[21] Barandela, R., & Juarez, M. (2001). Ongoing Learning for Supervised Pattern Recognition. Submitted to SIBGRAPI-2001, Brazil.

[22] Cover, T.M., & Hart, P.E. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, Volume IT-13, January, pages 21-27.

[23] Bruzzone, L., & 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.

[24] Eui-Hong (Sam), & Karypis, George (1999). Centroid-Based Document Classification: Analysis & Experimental Results.

[25] 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.

[26] Javidi, B., Zhang, G.S., & Li, J. (1997). Encrypted Optical Memory Using Double-random Phase Encoding. Appl. Opt. 36, 1054-1058.

[27] 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.

[28] Rueda, J.E., Romero, A.L., & 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. Disponible en: http://www.redalyc.org/articulo.oa?id=90330203

[29] Linfei, C., & Daomu, Z. (2005). Optical Image Encryption Based On Fractional Wavelet Transform, Opt. Comm. Vol. 254 (2005) p.p. 361-367.

[30] Hossam, E.A., Hamdy, K., & Osama, S.F.A. (2007). An Efficient CHAOS-BASED FEEDBACK STREAM CIPHER (ECBFSC) For Image Encryption And Decryption. Informática, volumen 3, pp. 121-129.

[31] Pisarchik, A.N., & 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.

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

[33] Rajan, B., & 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, abril, volumen 53 (número 4).

[34] 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. Disponible en: http://www.redalyc.org/articulo.oa?id=40440683002

[35] 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. Disponible en: http://www.redalyc.org/articulo.oa?id=613965326004

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

[37] 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. Disponible en: http://www.redalyc.org/articulo.oa?id=84923878015. URL: https://www.redalyc.org/articulo.oa?id=84923878015

[38] William, S. (1999). Cryptography and Network Security: Principles and Practice, 2nd edition, Prentice-Hall, Inc., pp 23-50.

[39] Progress Software Corporation, Telerik (2020-2022). 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/

[40] Luciano, D., & Prichett, G. (1987). Cryptology: From Caesar Ciphers To Public-key Cryptosystems. The College Mathematics Journal, vol 18 pp 2-17.

[41] Barandela, R., Sánchez, JS., García, V., & Rangel, E. (2003). Strategies for Learning in Class Imbalance Problems. Pattern Recognition, Vol. 36, No. 3 , pp. 849-851, 2003. Rapid and Brief Comunication (Pergamon) ISBN: (PII: S0031-3203(02)00257-1. 0031-3203/02/).

[42] Barandela, R., Sánchez, J.S., García, V., & Rangel, E. (2001b). Fusion of techniques for handling the imbalanced training sample problem. In: Procedings of 6th Ibero-American Symposium on Pattern Recognition, Brasil, 2001, 31-40.

[43] Hart, P.E. (1968). The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, 6,4,515-516, Vol. IT-14, No. 3, May.

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

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

[46] PyPI (2024). "Pycryptodome 3.21.0". Python Software Foundation. Retrieved from: https://pypi.org/project/pycryptodome/, (13/12/2024).