UACJ INSPIRA EL FUTURO

IIT

CULCYT

CULTURA CIENTÍFICA Y TECNOLÓGICA • ISSN 2007-0411

VOLUMEN 22 | NÚMERO 3 SEPTIEMBRE - DICIEMBRE 2025

CULCYT. CULTURA CIENTÍFICA Y TECNOLÓGICA
Revista de investigación en ingeniería e innovación tecnológica | ISSN 2007-0411

CULCYT. CULTURA CIENTÍFICA Y TECNOLÓGICA
Revista de investigación en ingeniería e innovación tecnológica | ISSN 2007-0411

Universidad Autónoma de Ciudad Juárez | Instituto de Ingeniería y Tecnología

CULCYT se fundó en diciembre de 2003 como parte del programa para la formación de investigadores del Instituto de Ingeniería y Tecnología. Lanzó su primer número en abril de 2004. Fundador: Dr. Victoriano Garza Almanza.

CULCYT. CULTURA CIENTÍFICA Y TECNOLÓGICA

Volumen 22, número 3, septiembre-diciembre 2025, es una publicación académica, editada de manera cuatrimestral por la Universidad Autónoma de Ciudad Juárez, a través del Instituto de Ingeniería y Tecnología.

D.R. © UACJ, Avenida del Charro núm. 619 norte, col. Partido Romero, C. P. 32310, Ciudad Juárez, Chihuahua, México.

Editor responsable: Dr. Manuel Antonio Ramos Murillo

Reserva de derechos al uso exclusivo núm. 04-2022-101711332100-102.

ISSN (electrónico): 2007-0411.

Sitio OJS: erevistas.uacj.mx/ojs/index.php/culcyt. Correo electrónico: culcyt@uacj.mx. Teléfono +52 (656) 6884848.

DOI: 10.20983/culcyt.2025.3

Modalidad de publicación continua. Tan pronto como un artículo ha sido preparado, se publica en línea.

Los trabajos a publicar en CULCYT deben ser originales e inéditos. En este momento, la revista no tiene costos de publicación para los autores.

El acceso a la revista es libre, sin requerimientos, bajo lo establecido en la normatividad mexicana de acceso abierto, y se da a través de su sitio

https://erevistas.uacj.mx/ojs/index.php/culcyt

o mediante el Repositorio Institucional de la UACJ

http://ri.uacj.mx/vufind/

Los artículos firmados son responsabilidad de sus autores.

Información para autores:

https://erevistas.uacj.mx/ojs/index.php/culcyt/autores

Contenido
Contents

Artículos de investigación
Research articles

Las ediciones de la revista CULCYT Cultura Científica y Tecnológica se publican cada cuatro meses, en la modalidad de publicación continua. Los trabajos se pueden redactar en idioma español o inglés, los cuales se someterán a una prueba de plagio utilizando herramientas electrónicas y a un proceso de revisión por pares doble ciega. La evaluación de cada trabajo puede ser realizada por hasta tres revisores acreditados de reconocimiento nacional e internacional. Cada revisor presentará una apreciación sobre la novedad, originalidad y la calidad del trabajo, y también evaluará el cumplimiento de las normas fijadas por el comité editorial referentes a las políticas editoriales y al formato de presentación de los manuscritos.

Los tipos de publicaciones que acepta CULCYT son artículos de investigación, artículos de revisión, notas de información técnica, cartas al editor o trabajos de excelencia que hayan sido galardonados con premios en congresos nacionales o internacionales.

PREPARACIÓN DEL MANUSCRITO

Todo manuscrito deberá estar redactado conforme a la plantilla proporcionada por CULCYT. Descargar aquí la plantilla del manuscrito.

Todo manuscrito deberá cumplir en su totalidad con las especificaciones de formato generales emitidas por CULCYT.

Se recomienda revisar la página Acerca de la revista para consultar las políticas de sección de CULCYT, así como la Guía para autores/as. Los autores/as deben registrarse en la revista antes de publicar o, si ya están registrados, pueden simplemente iniciar sesión y comenzar el proceso de cinco pasos.

El autor deberá cargar la Carta de Postulación llenada, firmada y convertida en formato PDF como archivo complementario. Descargar aquí la plantilla de la carta.

PLAZO DE LA PUBLICACIÓN

Por lo general, un artículo puede ser publicado en un tiempo promedio de tres meses. Este tiempo puede acortarse si el artículo cumple con los estándares de calidad de la publicación científica, sin embargo, en caso contrario puede alargarse.

- Algunas causas de que se alargue el tiempo de publicación son las siguientes:
- Incumplimiento de una o más de las normas de CULCYT establecidas en la Guía para Autores, lo cual motiva que el manuscrito se envíe a corrección preliminar antes de ser admitido para el inicio de la etapa de revisión por pares.
- Rechazo, falta de respuesta de revisores a la invitación de la revista o incumplimiento de los plazos de evaluación.
- Escasa calidad del manuscrito de forma tal que se tiene que enviar a más de una ronda de revisión por pares.
- Retrasos de los autores cuando se les solicitan correcciones.

CULCYT. Cultura Científica y Tecnológica
Vol. 22 | No. 3 | September-December 2025 | PP 5-13

DOI: 10.20983/culcyt.2025.3.2.1

ISSN (electronic) 2007-0411

e22301

RESEARCH ARTICLE

Optimality Criteria Optimization of Truss Structures Under Multiple Frequency Constraints by the Linear Approximation Resizing Rule

Optimización mediante el criterio de optimalidad de estructuras de barras bajo restricciones de múltiples frecuencias por la aproximación de la regla lineal de redimensionado

José Alfredo Ramírez Monares[1] ✉, *Elva Lilia Jardón Reynoso*[2], *Quirino Estrada Barbosa*[3]

[1] Ingeniería en Sistemas Automotrices, Departamento de Ingeniería Industrial y Manufactura, Campus Ciudad Universitaria, Universidad Autónoma de Ciudad Juárez, México

[2] Ingeniería en Manufactura, Departamento de Ingeniería Industrial y Manufactura, Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez, México

[3] Ingeniería en Diseño y Automatización Agrícola, Departamento de Ingeniería Industrial y Manufactura, Campus Nuevo Casas Grandes, Universidad Autónoma de Ciudad Juárez, México

ABSTRACT

The optimization of structures requires an efficient method to minimize weight, while satisfying multiple types of constraints. This approach generalizes the optimality criteria for the specific type of constraints in the frequency. Equations of motion for truss structures are considered to obtain the derivatives of the constraints required by the optimality criterion. Exponential and linear resizing optimization rules for the design variables are described. In the first, the optimized areas are compared with the analytical solution for a continuous rod. As a second example the optimized frequencies, weights and areas obtained by the linear resizing rule are compared to reference values. Both examples demonstrate the validity and effectiveness of the optimality criteria approach for the frequency constraints in truss structures.

KEYWORDS: optimization; optimality criterion; structural design; linear approximation.

RESUMEN

La optimización de estructuras requiere un método eficiente para minimizar el peso, a la vez que satisface múltiples tipos de restricciones. El presente enfoque consiste en generalizar los criterios de optimalidad para el tipo específico de restricciones en la frecuencia. Se tienen en cuenta las ecuaciones de movimiento de las estructuras de celosía para obtener las derivadas de las restricciones requeridas por el criterio de optimalidad. Se describen las reglas de optimización de redimensionamiento exponencial y lineal para las variables de diseño. En el primer ejemplo, las áreas optimizadas se comparan con la solución analítica como una varilla continua. En el segundo ejemplo, las frecuencias, pesos y áreas optimizados obtenidos por la regla de redimensionamiento lineal se comparan con las referencias. Ambos ejemplos demuestran la validez y eficacia del enfoque de criterios de optimalidad para las restricciones de frecuencia en estructuras de armaduras.

PALABRAS CLAVE: optimización; criterio de optimalidad; diseño estructural; aproximación lineal.

Corresponding author:
NAME: José Alfredo Ramírez Monares
INSTITUTION: Universidad Autónoma de Ciudad Juárez / Campus Ciudad Universitaria
ADDRESS: Av. Plutarco Elías Calles núm. 1210, Fovissste Chamizal, C. P. 32310, Ciudad Juárez, Chihuahua, México
E-MAIL: jose.ramirez@uacj.mx

Received: 8 September 2025. Accepted: 11 November 2025.
Published: 28 November 2025.

open access UACJ

PEER REVIEWED

I. INTRODUCTION

Research on practical applications of the Optimality Criteria in truss structures under frequency restrictions by the linear approximation resizing rule has emerged as a critical area of inquiry due to its relevance in enhancing structural performance while ensuring dynamic stability [1], [2]. The evolution of this field spans from early topology optimization methods focusing on static criteria to advanced multi-criteria approaches incorporating natural frequency constraints and dynamic responses [3], [4]. Practical significance is underscored by the widespread use of lattice and truss structures in aerospace, civil, and mechanical engineering, where vibration control is essential to prevent resonance-induced failures [5], [6]. Studies report that optimized lattice structures can achieve significant weight reduction while increasing fundamental frequencies, directly impacting safety and efficiency [7], [8].

The specific problem addressed involves optimizing truss structures to satisfy frequency constraints [9], [10]. Despite advances, a notable knowledge gap persists in effectively integrating optimality criteria methods with linear approximation resizing rules under frequency constraints, especially in practical engineering applications [3], [11], [12]. Controversies arise regarding the best optimization algorithms; metaheuristic versus gradient-based and the handling of multiple frequency constraints and mode switching [10], [13], [14]. Failure to resolve these issues leads to suboptimal designs prone to dynamic instabilities and increased computational costs [15], [16].

Topology optimization defines material distribution, frequency constraints ensure dynamic performance, and linear approximation resizing facilitates efficient iterative updates. This framework guides the systematic review to evaluate how these concepts coalesce in practical truss structure design, grounded in structural dynamics and optimization theory [17], [18].

It is often important for structures subjected to dynamic loading that some frequencies are higher by some prescribed margin than the predominant frequencies. Furthermore, by using the well-developed optimality criterion approaches for design and efficient procedures for the implicit requirement of natural frequency analysis, the computational effort can be limited to within the same order of magnitude as that required for static loads analysis. This paper describes an effective optimality criterion computer design approach for member size selection to fulfill frequencies for truss structures. Optimality criteria methods use a relation that is based on the condition that a design is expected to meet at its optimum when the gradient for the frequency constraint is equal to the gradient of the objective function. Namely the weighted sum of the Lagrangian energy densities corresponding to multiple frequency constraints should be equal to unity in all the elements.

There are several works about the optimality criteria method in truss structures. For example, in [19] there are shown some optimality criterion algorithms for different constraints; however, such algorithms depend on nine control parameters, unlike the alone step-size parameter η used in the linear approximation. In [20], the linear approximation of the optimality criterion is applied, but only with stress and displacement constraints under static loading conditions.

The objective here is to develop an optimality criterion method for multiple constraints on the frequencies; that method should quickly reduce the weight of the starting design and converge to the optimal design variables. [21] and [22] have extended the method to multiple frequency constraints. This paper addresses the minimization of the structural mass in truss structures subject to frequency constraints under free vibration conditions. It differs from the works of [21], [22] and [23] in that the design scaling follows a different rule, so the use of the weighting parameter has been avoided. Just the step-size parameter is used.

II. METHODOLOGY

2.1. FREQUENCY ANALYSIS

The natural frequency analysis for a free undamped vibration of a discretized structure consists of finding a solution to the homogeneous set of a n-th order matrix represented by

$$[K]\psi_j - \omega_j^2[M + \overline{M}]\psi_j = 0 \qquad (1)$$
$$j = 1, 2, \ldots, n$$

where $[K]$, $[M]$, and $[\overline{M}]$ are the total stiffness, mass, and nonstructural mass matrices; ω_j is the circular frequency associated with the j-th vibration mode ψ_j. Multiplying Equation (1) by ψ_j^T gives

$$\psi_j^T[K]\psi_j - \omega_j^2\psi_j^T[M + \overline{M}]\psi_j = 0 \qquad (2)$$

DOI: 10.20983/culcyt.2025.3.2.1

Thus, solving for the circular frequency ω_j^2 we get

$$\omega_j^2 = \frac{(\psi_j^T[K]\psi_j)}{(\psi_j^T[M+\overline{M}]\psi_j)} \tag{3}$$

which is the Rayleigh quotient. The gradient of the circular frequency with respect to the design variable A_i is obtained by differentiating Equation (2); this gives

$$\frac{\partial\omega_j^2}{\partial A_i} = \frac{1}{A_i}\,\frac{(\psi_{j_i}^T[k]_i\psi_{j_i} - \omega_j^2\psi_{j_i}^T[m]_i\psi_{j_i})}{(\psi_j^T[M+\overline{M}]\psi_j)} \tag{4}$$

where ψ_{j_i} is the component of the vibration mode associated with the i-th element A_i, $[k]_i$, and $[m]_i$ are the stiffness and mass matrices of the i-th element. In deriving Equation (4) it was assumed that the stiffness and mass matrices are linear functions of the design variable A_i. According to [24], the stiffness and the consistent mass matrices of a bar are, respectively.

$$[K] = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5}$$

$$[M]_c = \frac{\rho AL}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{6}$$

In this manner, the derivatives of Equation (4) are valid for finite elements whose stiffness and mass matrices are linear functions of the area, such as demonstrated in Equations (5) and (6) for bars. Normalizing with respect to the mass matrix $[M+\overline{M}]$, Equation (4) is written as

$$\frac{\partial\omega_j^2}{\partial A_i} = \frac{1}{A_i}\,\{\tilde{\psi}_j\}_i^T[k]_i\{\tilde{\psi}_j\}_i - \omega_j^2\{\tilde{\psi}_j\}_i^T[m]_i\{\tilde{\psi}_j\}_i \tag{7}$$

where

$$\{\tilde{\psi}_j\}_i = \frac{\{\psi_j\}}{\left[\{\psi_j\}^T[M+\overline{M}]\{\psi_j\}\right]^{\frac{1}{2}}} \tag{8}$$

From now on, Equation (8) is being used in the forthcoming optimality criterion.

2.2. OPTIMIZATION PROCEDURE

The optimization problem is defined as minimizing the structural weight.

$$W(x_i) = \rho_i l_i x_i \qquad i = 1, 2, \ldots, n \tag{9}$$

subject to m constraints

$$g_j(x_i) = \omega_i^2 - \overline{\omega}_j^2 = 0 \qquad j = 1, 2, \ldots, k \tag{10}$$

$$g_j(x_i) = \omega_i^2 - \overline{\omega}_j^2 \leq 0 \qquad j = k+1, k+2, \ldots, m \tag{11}$$

where ρ_i is the mass density, x_i is the design variable, and l_i is the length of the i element. In Equations (10) and (11), ω_j and $\overline{\omega}_j$ are the actual and desired values of the frequency constraints. In addition, minimum limits are prescribed on the design variables $x_i > x_i^l$.

Using Equations (9), (10) and (11), the Lagrangian function, L, as

$$L(x_i, \lambda) = \rho_i l_i x_i - \lambda_j(\omega_j^2 - \overline{\omega}_j^2) \qquad i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m \tag{12}$$

where λ_j's are the Lagrange multipliers. Differentiating this equation with respect to the design variables and setting the resulting equations to zero, the optimality criterion leads to

$$\frac{\partial W(x_i)}{\partial A_i} - \sum_{j=1}^{m}\lambda_j\frac{\partial\omega_j^2}{\partial A_i} = 0 \tag{13}$$

i.e.

$$\rho_i l_i - \sum_{j=1}^{m}\lambda_j\frac{\partial\omega_j^2}{\partial A_i} = 0 \tag{14}$$

Thus, Lagrangian energy density is expressed as

$$\sum_{j=1}^{m}\frac{\lambda_j}{\rho_i l_i}\frac{\partial\omega_j^2}{\partial A_i} = 1 \tag{15}$$

Substituting the expression of the frequency derivative given by Equation (7) in Equation (15) yields

$$e_{ij} = \frac{\lambda_j\left[\{\tilde{\psi}_j\}_i^T[k]_i\{\tilde{\psi}_j\}_i - \omega_j^2\{\tilde{\psi}_j\}_i^T[m]_i\{\tilde{\psi}_j\}_i\right]}{A_i\rho_i l_i} = 1 \tag{16}$$

This represents the ratio of the gradient for the frequency constraint to the gradient of the objective function.

Now it is possible to write recursive relations to determine the Lagrange multipliers and modify the design variables. In both cases, these recursive relations are written in an exponential or linearized form. In the works of [21], [22], and [23] the exponential recursive relations are used, so that the design variables are modified by multiplying them by a quantity which is equal to unity at the optimum. The exponential rule is given by

$$x_i^{\text{new}} = x_i \left(\frac{1}{x_i^2 f_i} \sum_{j=1}^{ng} \lambda_j c_{ij} \right)^{1/\eta} \qquad (17)$$
$$i = 1, 2, \ldots, n$$

where

$$f_i = \frac{\partial W(x_i)}{\partial x_i}, \qquad c_{ij} = -x_i^2 \frac{\partial g_j}{\partial x_i} \qquad (18)$$
$$i = 1, 2, \ldots, n$$

In the present work, the linear recursive relation is used for the estimation of the design variables, where they are modified by adding a quantity, Δx_i, which is equal to zero at the optimum. A linearized form of the Equation (17), obtained by binomial expansion, is

$$x_i^{\text{new}} = x_i + \Delta x_i \qquad (19)$$
$$i = 1, 2, \ldots, n$$

where

$$\Delta x_i = \frac{1}{\eta} \left(\frac{1}{x_i^2 f_i} \sum_{j=1}^{m} \lambda_j c_{ij} - 1 \right) x_i \qquad (20)$$
$$i = 1, 2, \ldots, n$$

and

$$\sum_{j=1}^{m} \sum_{i=1}^{n} \frac{c_{il} c_{ij}}{x_i^3 f_i} \lambda_j = \sum_{i=1}^{n} \frac{c_{il}}{x_i} \eta g_l(x_i) \qquad (21)$$
$$i = 1, 2, \ldots, m$$

The term η is a step size parameter. The old value for x_i is used to produce a new estimate. Note that the linear recursive relation for the Lagrange multipliers, Equation (21), approximates a set of linear equations that can be used to determine the Lagrange multipliers λ_j. Nonetheless, it is possible to investigate the convergence of these linear relations and compare them with the exponential relation results presented by the references.

III. RESULTS

3.1. ROD WITH A TIP MASS

Figure 1 shows the mechanical system analyzed herein presented by [25] and [23]. It is a continuous one-dimensional rod with a tip mass M_c = 0.1 lb s^2/in. The rod is discretized in ten bar elements with equal length l_i = 9 in i = 1, …, 10. The fundamental axial frequency of the system is given a specified value, $\omega_1 \geq 2576.1$ rad/s = 410 Hz, and the sectional areas of the bars, A_i i = 1, …, 10, are the design variables required for which the total mass is a minimum. The rod is built from the linearly elastic material featuring modulus of elasticity E = 10.3 psi and density $\rho = 2.59\times10^{-4}$ lb/in^3. The bar has ten degrees of freedom for joint translation in the horizontal direction, $u(y, t)$. According to [25], the analytical solution of the optimization problem constrained by the fundamental frequency leads to

$$A(x) = \frac{m(L)\cosh^2(\beta_1 L)}{\rho \cosh^2(\beta_1 y)} \qquad (22)$$

where

$$m(L) = \beta_1 M_c \tanh(\beta_1 L) \qquad (23)$$

and

$$\beta_1 = \omega \sqrt{(\rho / E)} \qquad (24)$$

β_1 represents the frequency parameter associated with the fundamental mode.

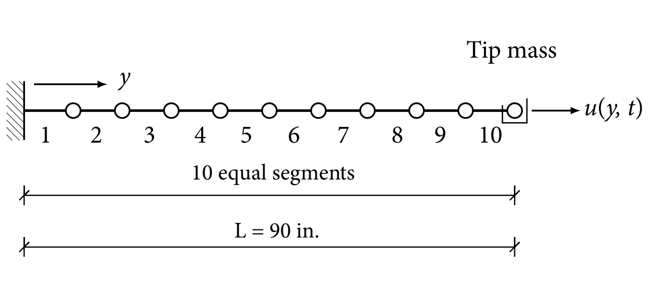

Figure 1. Ten bars structure.

Figure 2 shows the optimized cross-sectional areas obtained by both the analytical solution [25] and the here obtained optimization scheme for the discretized model. There is a difference that increases in the segments located at the left and right endpoints. In the left endpoint the optimization scheme underestimates the areas of the analytical function; in the right endpoint it overestimates these areas. A bigger match between both methods is observed in the range ($30 \leq y \leq 50$) in.

Figure 3 a) shows the optimization history for the first 10 iterations of the fundamental frequency here obtained. The starting values in all the areas of the bars are 9.3 in^2 for both the work of [23] and the present work. Figure 3 b) shows the weight iteration history. The linear optimization scheme exhibits a quick reduction of the starting weight and a quite fast convergence to the optimum values. The weight of the optimized structure by the analytical scheme is 80.44 lbs. The work of [23] reports a weight of 81.2 lbs. The weight of the optimized rod here obtained is 80.96 lbs., a lighter weight structure closer to the analytical weight value, despite the differences in the values of the area for the leftmost and the rightmost extremes shown in Figure 2.

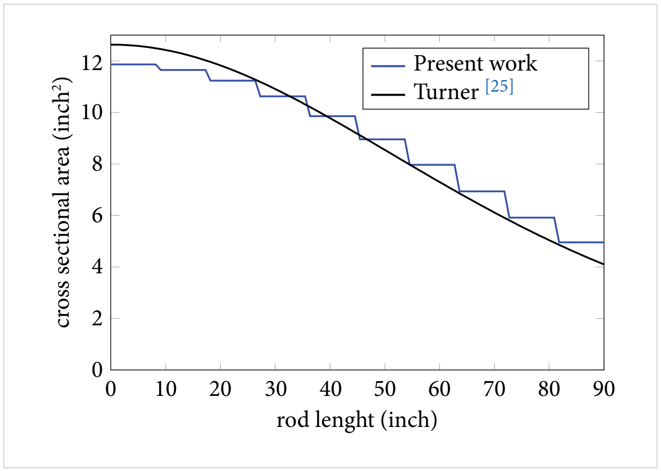

Figure 2. Optimal area distributions in the rod.

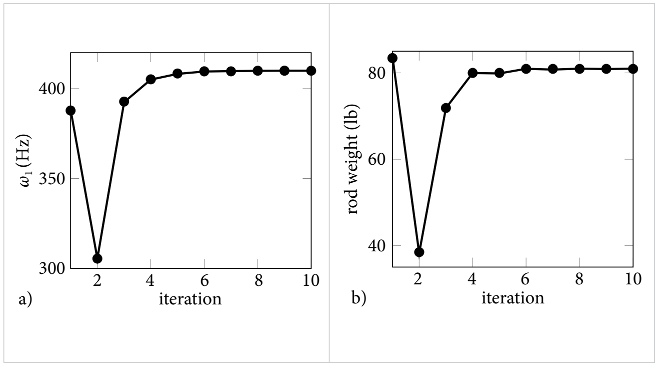

Figure 3. Iteration history: a) frequency history for ω_1 and b) rod weight history.

3.2. TEN BAR TRUSS

Figure 4 shows the ten-bar truss analyzed herein presented by [21]. It is a two-dimensional truss structure with tip masses $M_c = 2.588$ lb s²/inch at the nodes 1, ..., 4. The structure is discretized in 10 bar elements from the linear elastic material featuring modulus of elasticity $E = 1{\times}10^7$ psi and density $\rho = 0.1$ lb/in³. The axial frequencies of the system are given several specified values, and the sectional areas of the bars, A_i $i = 1, \ldots, 10$, are the design variables required for which the total mass is a minimum. There is a lower limit $x_l = 0.1$ for the design variables.

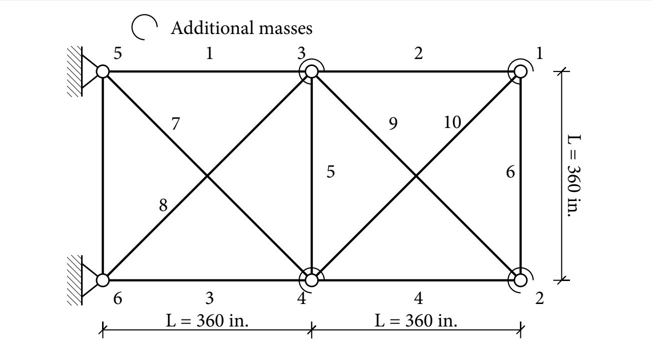

Figure 4. Ten-bar truss structure.

The optimization scheme utilizes initial values of 9.5318 in² across all bar areas for both, reference [21] and the present work. The second frequency must be 10, 15, 20, 25, 27.08, and 30 Hz across six distinct scenarios. Table 1 presents the optimal design frequencies and weights under specified constraints. The lines "%" show the absolute percentage difference between the reference values [21] and those obtained here. Only in the initial design are equal weights of 4000 lbs. for both, reference [21] and present work. A lighter structure results from the linear approximation rule in other restrictions. The major difference between both occurs at the restriction $\omega_2 = 30$ Hz, where the present work attains a 26.15% lighter weight than the reported by the reference [21].

TABLE 1
Ten-Bar Truss Second Frequency Values (Hz) Obtained by [21] and Present Work (p.w.)

ω^2	Ref.	Frequency number								Weight (lbs)
		1	2	3	4	5	6	7	8	
Initial design	[21]	8.96	27.08	27.45	51.25	58	64.73	66.87	80.85	4000
	p.w.	9.177	27.31	29.78	53.8	61.05	68.34	69.93	82.09	4000
	%	2.42	0.85	8.49	4.98	5.26	5.58	4.58	1.53	0
10	[21]	3.26	10	10.19	16.01	18.08	22.96	25.21	27.25	304.5
	p.w.	3.179	9.992	10.16	13.36	13.92	19.33	25.13	26.28	269.45
	%	2.5	0.08	0.3	18.17	25.79	17.15	0.32	3.68	12.28
15	[21]	4.92	15.0	15.07	15.3	22.21	24.28	39.49	41.64	637.0
	p.w.	4.762	14.98	15.19	18.27	20.35	28.27	38.08	39.59	616.05
	%	3.27	0.13	0.79	17.52	8.76	15.28	3.65	4.97	3.36
20	[21]	6.64	20.0	20.13	21.51	30.39	32.81	52.53	55.89	1251.5
	p.w.	6.386	19.93	20.19	23.91	27.22	36.98	50.38	52.46	1158.4
	%	3.9	0.35	0.3	10.58	11.08	11.95	4.21	6.3	7.71
25	[21]	8.4	25	25.0	27.66	39.34	41.32	65.35	70.49	2243.8
	p.w.	8.095	24.91	25.23	28.93	34.73	44.81	62.02	65.08	1970.3
	%	3.7	0.36	0.92	4.48	12.44	8.15	5.23	8.07	12.87
27.8	[21]	9.17	27.08	27.11	30.96	43.96	45.54	70.4	76.84	2865.9
	p.w.	8.825	27.06	27.37	31.57	37.46	48.55	67.64	70.79	2424
	%	3.84	0.07	0.95	1.96	16.09	6.4	4.01	8.19	16.71
30	[21]	10.34	30.0	30.13	35.4	50.73	51.39	78.9	87.71	4143.9
	p.w.	9.867	29.98	30.34	34.76	41.68	53.28	74.42	78.06	3189.2
	%	4.68	0.07	0.69	1.83	19.34	3.64	5.85	12.28	26.15

The optimized area values are shown in Table 2. Despite the differences in the optimum design values, both results fulfill the frequency restrictions. There are no optimum variables at the lower limit x^l for this restriction.

TABLE 2
Ten-Bar Truss Area Values (inch²) Obtained by [21] and Present Work (p.w.)

ω^2	Ref.	Optimum design variables									
		1	2	3	4	5	6	7	8	9	10
10	[21]	0. 91	0.821	0.91	0.821	0.768	0.57	0.712	0.712	0.581	0.581
	p.w.	0.903	0.795	0.903	0.795	0.457	0.23	0.781	0.781	0.362	0.362
	%	0.77	3.22	0.77	3.22	50.6	82.35	9.25	9.25	46.81	46.81
15	[21]	2.313	2.154	3.313	2.154	0.602	0.353	1.723	1.723	1.037	1.036
	p.w.	2.23	1.888	2.23	1.888	1.098	0.534	1.785	1.785	0.793	0.793
	%	3.67	13.14	38.4	13.14	58.21	40.85	3.54	3.54	26.82	26.86
20	[21]	4.435	4.14	4.435	4.14	1.223	0.76	3.413	3.413	2.114	2.114
	p.w.	4.228	3.488	4.228	3.488	1.998	0.934	3.425	3.425	1.476	1.476
	%	4.78	17.15	4.78	17.15	48.71	20.61	0.35	0.35	35.34	35.34
25	[21]	7.699	7.224	7.697	7.223	2.195	1.382	6.211	6.211	4.017	4.003
	p.w.	7.038	5.708	7.038	5.708	3.07	1.394	6.11	6.11	2.637	2.637
	%	9.01	23.51	9	23.5	33.19	0.87	1.64	1.64	41.78	41.3
27.8	[21]	9.598	9.979	9.598	8.979	2.905	1.85	7.898	7.898	5.431	5.431
	p.w.	8.985	7.049	.985	7.049	3.896	1.703	7.428	7.428	3.057	3.057
	%	6.66	34.33	6.66	23.94	29.13	8.3	6.16	6.16	55.98	55.98
30	[21]	13.72	12.86	13.72	12.86	3.907	2.774	11.07	11.07	8.46	8.46
	p.w.	11.91	9.047	11.91	9.047	5.069	2.124	9.989	9.989	3.95	3.95
	%	14.19	34.78	14.19	34.78	25.87	26.65	10.2	10.2	71.86	71.86

The design of the ten-bar truss has now been developed, considering arrays of constraints pertaining to various frequency considerations that must be adhered to ensure optimal structural performance. The minimization of the structural weight is executed in such a manner that the fundamental frequency, along with the second and third frequencies, is subjected to specific constraints that ensure their values remain within predetermined limits.

The first set is with ω_1 = 7.0 Hz, the second with ω_1=10.0 Hz, the third with ω_1 = 7.0 Hz and $\omega_2 \geq$15 Hz, the fourth with ω_1 = 10.0 Hz and $\omega_2 \geq$ 15.0 Hz, the fifth with ω_1 = 7.0 Hz, $\omega_2 \geq$ 15.0 Hz and $\omega_3 \geq$ 20.0 Hz and, finally, all inequality constraints, $\omega_1 \geq$ 3.5 Hz, $\omega_2 \geq$ 10.0 Hz and $\omega_3 \geq$ 14.0 Hz were considered. Table 3 presents the results obtained under these constraint conditions. It is observed that the present work results in a heavier weight optimized structure than the reference only in the first set; it is just a 0.14% increase in weight. For all the other sets the opposite happens; the linear approximation here used attains lighter weight optimized structures. The major difference occurs in the sixth set; it is a 13.43% difference in weights.

TABLE 3
Ten-Bar Truss Second Frequency Values (Hz) Obtained by [21] and Present Work (p.w.)

Set	Ref.	Frequency number								Weight (lbs)
		1	2	3	4	5	6	7	8	
1	[21]	7.0	10.96	16.27	18.21	27.39	29.55	47.92	50.34	1137.3
	p.w.	7	11.29	16.6	19.47	28.33	31.13	47.52	50.84	1138.9
	%	0	2.97	2.01	6.7	3.37	5.27	0.84	0.99	0.14
2	[21]	10	13.73	22.29	25.19	38.04	42.21	65.79	69.93	2614.0
	p.w.	10	14.38	23.04	25.64	37.83	39.89	65.8	69.17	2526.3
	%	0	4.64	3.31	1.77	0.55	5.67	0.02	1.09	3.41
3	[21]	7.0	15.58	16.93	18.75	29.13	30.3	46.93	49.67	1172.6
	p.w.	7	15	16.76	18.48	27.87	28.79	47.96	50.17	1158.8
	%	0	3.79	1.01	1.45	4.4	5.08	2.17	1	1.18
4	[21]	10	19.16	24.52	27.16	38.71	40.53	67.66	71.38	2736.3
	p.w.	10	15	23.29	27.23	39.45	43.15	64.95	70.06	2557.3
	%	0	24.57	5.14	0.26	1.9	6.22	4.08	1.86	6.76
5	[21]	7.0	15.61	20.17	20.77	28.76	29.76	53.88	56.03	1308.4
	p.w.	7	15.0	20.0	20.22	29.63	34.3	46.75	52.31	1243.2
	%	0	3.99	0.85	2.7	2.97	14.07	14.28	6.9	5.05
6	[21]	4.4	12.14	14.0	17.89	19.58	22.96	34.01	35.72	489.17
	p.w.	3.5	10.0	14.0	16.02	17.26	23.77	27.91	34.13	427.8
	%	22.65	19.57	0	11.02	12.5	3.46	19.86	4.55	13.43

Table 4 consists of all the design variables at the optimum. The variables A_5 and A_6 became passive after eight iterations for the first set. The variables A_5 and A_6 became passive after the third and ninth iterations respectively for the second set.

TABLE 4
Ten-Bar Truss Frequency Values (Hz) Obtained by [21] and Present Work (p.w.)

Set	Ref.	Optimum design variables									
		1	2	3	4	5	6	7	8	9	10
1	[21]	6.045	1.969	6.045	1.969	0.1	0.1	3.206	3.206	2.226	2.226
	p.w.	5.228	2.212	5.228	2.212	0.1	0.1	3.418	3.418	2.434	2.434
	%	14.5	11.66	14.5	11.66	0	0	6.44	6.44	8.92	8.92
2	[21]	13.965	4.437	13.965	4.437	0.1	0.1	7.579	7.579	5.009	5.009
	p.w.	12.176	4.503	12.176	4.503	0.1	0.1	7.93	7.93	5.014	5.014
	%	13.78	1.47	13.78	1.47	0	0	4.52	4.52	0.1	0.1
3	[21]	5.511	1.937	5.511	1.937	0.207	0.414	3.616	3.616	2.414	2.414
	p.w.	5.251	2.222	5.251	2.222	0.299	0.298	3.438	3.438	2.446	2.446
	%	4.84	13.76	4.84	13.76	37.07	32.89	5.08	5.08	1.32	1.32
4	[21]	13.147	5.683	13.147	5.683	0.488	0.517	9.093	9.093	4.11	4.11
	p.w.	12.178	4.541	12.178	4.541	0.326	0.111	8.079	8.079	5.058	5.058
	%	7.69	22.25	7.69	22.25	39.75	128.42	11.75	11.75	20.59	20.59
5	[21]	5.672	3.823	5.672	3.823	0.646	0.321	4.191	4.191	1.604	1.604
	p.w.	6.039	0.819	5.46	2.875	0.871	0.584	3.417	3.83	3.247	2.126
	%	6.26	134.18	3.83	28.32	29.7	57.75	20.4	8.92	67.09	27.6
6	[21]	2.306	1.304	2.306	1.304	0.639	0.557	1.029	1.029	0.8	0.8
	p.w.	0.749	0.514	0.98	1.571	1.077	0.408	2.103	1.17	0.687	0.692
	%	104.91	85.39	80.08	18.78	51.27	30.73	68.21	12.98	15.22	14.54

The linear approximation scheme here presented attains lighter weight structures in almost all cases of restrictions on the first, second and third frequencies; it was demonstrated how the linear approximation here used can effectively optimize the truss structures here studied. Optimization of the structure to attain specific frequencies can be achieved by using the linear approximation, which is a function of the stiffness, the structural and the non-structural masses.

V. CONCLUSIONS

This study has demonstrated that the application of optimality criteria, particularly through the linear resizing rule, provides an effective strategy for addressing structural optimization problems under multiple frequency constraints in discretized truss systems. Across the examples analyzed, the proposed formulation not only achieved significant weight reductions but also ensured compliance with the imposed dynamic conditions, while maintaining consistency with reference results reported in the literature.

In the rod with a tip mass example, the optimized cross-sectional areas showed an elevated level of agreement with the continuous analytical solution, confirming the robustness of the method when applied to discretized models. Although slight deviations were observed at the ends of the bar, the linear approximation produced a final weight closer to the analytical value, which highlights its ability to converge rapidly toward optimal designs without compromising accuracy.

Similarly, in the ten-bar truss case, the method once again proved effective by yielding solutions that, in most scenarios, resulted in lighter structures than those obtained in previous studies, while still fulfilling all frequency requirements. It is worth noting that weight reductions were particularly significant under more demanding constraint conditions, which underscores the advantages of the linear approximation over traditional schemes based on exponential rules or multiple control parameters.

A noteworthy aspect of this approach is that the linear resizing rule relies solely on a single step-size parameter, thereby avoiding the need for additional weighting factors. This not only simplifies its implementation but also enhances its adaptability to different structural types and design scenarios, broadening its potential for practical applications.

Overall, the results confirm that the proposed method achieves a favorable balance between computational efficiency, operational simplicity, and reliability of optimized designs. These attributes position it as a solid alternative for structural optimization under dynamic constraints, in comparison with other approaches that, although effective, typically demand greater parametrization and higher computational effort.

Finally, it is emphasized that, while the case studies confirm the validity of the method, future research may extend its application to problems involving nonlinear dynamic loads, damping effects, and three-dimensional structures. Advancing in these directions would further consolidate the linear approximation of optimality criteria as a reliable tool for the advanced design of lightweight and efficient structures subject to dynamic performance requirements.

REFERENCES

[1] V. Savsani, G. Tejani, and V. Patel, "Topology, Shape, and Size Optimization," in *Truss Optimization*, V. Savsani, G, Tejani, and V. Patel, Eds. Switzerland: Springer Nature, 2024, pp. 241–359, doi: 10.1007/978-3-031-49295-2_6.

[2] L. Siqueira, E. Silva, and R. Picelli, "Structural Topology Optimization with Volume and Natural Frequency Constraints by Using the TOBS Method," in *Proceedings of the 8th International Symposium on Solid Mechanics*, M. Bittencourt and J. Labaki, Eds. 2024, pp. 79–92, doi: 10.1007/978-3-031-59804-3_5.

[3] Q. Peng, T. Lin, W. Liu, and B. Chen, "An optimality criteria method hybridized with dual programming for topology optimization under multiple constraints by moving asymptotes approximation," *Comput Mech*, vol. 69, no. 3, pp. 683–699, Mar. 2022, doi: 10.1007/s00466-021-02110-5.

[4] K. Liu, Y. Bai, S. Yao, and S. Luan, "Topology optimization of shell-infill structures for natural frequencies," *Eng Comput*, vol. 39, no. 8, pp. 3083–3107, Aug. 2022, doi: 10.1108/EC-03-2022-0135.

[5] Z. Wu, J. Wu, F. Lu, C. Zhang, Z. Liu, and Y. Zhu, "Free vibration analysis and multi-objective optimization of lattice sandwich beams," *Mech. Adv. Mater. Struct.*, vol. 31, no. 17, pp. 4037–4050, Sep. 2024, doi: 10.1080/15376494.2023.2189333.

[6] C. Wang, B. Zhang, S. Huang, W. Dou, S. Xin, and J. Yan, "Topological Design of a Nanosatellite Structure with Optimal Frequency Responses Filled by Non-Uniform Lattices," *Chin. J. Mech. Eng.*, vol. 37, no. 1, p. 161, Dec. 2024, doi: 10.1186/s10033-024-01156-9.

[7] L. Chen, Y. Pan, X. Chu, H. Liu, and X. Wang, "Multiscale design and experimental verification of Voronoi graded stochastic lattice structures for the natural frequency maximization problem," *Acta Mechanica Sinica*, vol. 39, no. 8, p. 422445, Aug. 2023, doi: 10.1007/s10409-023-22445-x.

[8] M. Beghini *et al.*, "Tuning Modal Behavior of Additively Manufactured Lattice Structures," *J Eng Gas Turbine Power,* vol. 146, no. 7, Jul. 2024, doi: 10.1115/1.4064264.

[9] A. A. Pessoa and J. M. Aroztegui, "A Cutting Plane Approach to Maximization of Fundamental Frequency in Truss Topology Optimization," *Research Square*, Oct. 24, 2023, doi: 10.21203/rs.3.rs-3459452/v1.

[10] V. Goodarzimehr, U. Topal, A. K. Das, and T. Vo-Duy, "SABO algorithm for optimum design of truss structures with multiple frequency constraints," *Mech. Based Des. Struct. Mach.*, vol. 52, no. 10, pp. 7745–7777, Oct. 2024, doi: 10.1080/15397734.2024.2308652.

[11] Z. Deng, Y. Liang, and G. Cheng, "Discrete variable topology optimization for maximizing single/multiple natural frequencies and frequency gaps considering the topological constraint," *Int J Numer Methods Eng*, vol. 125, no. 10, May 2024, doi: 10.1002/nme.7449.

[12] A. Kaveh and H. Yousefpoor, "Chaotically Enhanced Meta-Heuristic Algorithms for Optimal Design of Truss Structures with Frequency Constraints," *Period. Polytech. Civil Eng.*, vol. 66, no. 3, pp. 900–921, Jan. 2022, doi: 10.3311/PPci.20220.

[13] A. Kaveh and H. Yousefpour, "Comparison of Three Chaotic Meta-heuristic Algorithms for the Optimal Design of Truss Structures with Frequency Constraints," *Period. Polytech. Civil Eng.*, vol. 67, no. 4, Jan. 2023, doi: 10.3311/PPci.22594.

[14] X. Teng, Q. Li, and X. Jiang, "A Smooth Bidirectional Evolutionary Structural Optimization of Vibrational Structures for Natural Frequency and Dynamic Compliance," *Comput. Model. Eng. Sci.*, vol. 135, no. 3, pp. 2479–2496, 2023, doi: 10.32604/cmes.2023.023110.

[15] Q. Wu, Q. Li, and S. Liu, "A method for eliminating local modes caused by isolated structures in dynamic topology optimization," *Comput Methods Appl Mech Eng*, vol. 418, p. 116557, Jan. 2024, doi: 10.1016/j.cma.2023.116557.

[16] H. R. Najafabadi, T. C. Martins, J. Hanamoto, M. S. G. Tsuzuki, and A. Barari, "Natural Frequency Control Using Simulated Annealing-Based Binary Topology Optimization," *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*, Nov. 2023, pp. 1463–1467, doi: 10.1109/INDUSCON58041.2023.10374765.

[17] V. Shah, M. Pamwar, B. Sangha, and I. Y. Kim, "Multi-material topology optimization considering natural frequency constraint," *Eng Comput*, vol. 39, no. 7, pp. 2604–2629, Jul. 2022, doi: 10.1108/EC-07-2021-0421.

[18] W. Wei, T. Qingguo, W. Fengbin, F. Yesen, Z. Shikun, and Z. Wenhui, "A Multi-Objective Topology Optimization Method Used in Simultaneous Constraints of Natural Frequency and Static Stiffness," *2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)*, Hangzhou, China, Mar. 2022, pp. 7–12, doi: 10.1109/CACML55074.2022.00010.

[19] A. Zacharopoulos, K. D. Willmert, and M. R. Khan, "An optimality criterion method for structures with stress, displacement and frequency constraints," *Comput Struct*, vol. 19, no. 4, 1984, doi: 10.1016/0045-7949(84)90109-3.

[20] T. R. Haftka and G. Zafer, *Elements of Structural Optimization*, 3rd ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1991, doi: 10.1007/978-94-011-2550-5.

[21] R. V. Grandhi and V. B. Venkayya, "Structural optimization with frequency constraints," *AIAA Journal*, vol. 26, no. 7, pp. 858–866, 1988, doi: 10.2514/3.9979.

[22] R. Canfield, V. Venkayya, and R. Grandhi, "Structural Optimization with Stiffness and Frequency Constraints," *Mechanics of Structures and Machines*, vol. 17, no. 1, pp. 95–110, Mar. 1989, doi: 10.1080/089054508915631.

[23] R. Levy and K. Chai, "Implementation of natural frequency analysis and optimality criterion design," *Comput. Struct.*, vol. 10, nos. 1–2, pp. 277-282, Apr. 1979, doi: 10.1016/0045-7949(79)90096-8.

[24] G. Dasgupta, "Finite Element Basics with the Bar Element: Uniaxial Deformations—Interpolants, Stiffness Matrices and Nodal Loads," in *Finite Element Concepts*, G. Dasgupta, Ed. New York, NY: Springer New York, 2018, pp. 1–41, doi: 10.1007/978-1-4939-7423-8_1.

[25] M. J. Turner, "Design of minimum mass structures with specified natural frequencies," *AIAA Journal*, vol. 5, no. 3, pp. 406–412, 1967, doi: 10.2514/3.3994.

RESEARCH ARTICLE

CULCYT. Cultura Científica y Tecnológica
Vol. 22 | No. 3 | September-December 2025 | PP 14-33

DOI: 10.20983/culcyt.2025.3.2.2

CULCYT
ISSN (electronic) 2007-0411

e22302

ChaCha20 Encryption Algorithm Security Enhancement through Artificial Intelligence-Based Random Noisy Injection: A Case Study

Inyección de ruido con inteligencia artificial para mejorar la seguridad de datos: Un caso de estudio del algoritmo ChaCha20

Edgar Rangel Lugo[1a] ✉, *Kevin Uriel Rangel Ríos*[1a], *Leonel González Vidales*[1a], *Carlos Alberto Bernal Beltrán*[1c], *Cinthya Maybeth Rangel Ríos*[1a], *Rosa Isabel Reynoso Andrés*[1b], *César del Ángel Rodríguez Torres*[1a], *Lucero de Jesús Ascencio Antúnez*[1a]

[1] [a]{Departamento de Sistemas y Computación}, [b]{Departamento de Desarrollo Académico}, [c]{Subdirección Académica}, Tecnológico Nacional de México / Instituto Tecnológico de Ciudad Altamirano, Guerrero, México

ABSTRACT

The problem of digital data theft is receiving growing attention in organizations because it may produce significant financial losses. This issue can be handled using dynamic encryption methodologies. There exists safety encryption alternatives such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman). However, it is known that these algorithms have been threatened by quantum computing advent. Thereby, the aim of this research is to suggest novel dynamic encryption alternatives using artificial intelligence (AI), based on a noisy injection scheme on ciphertext, as it has the potential to mislead cybercriminals. Several aspects related to this subject were studied. Despite that quantum computing was not used, other measures have been proposed. The designed methodology was focused over the updating of ChaCha20 strategy combined with random Caesar II methodology. This fusion of techniques, referred to as random noisy ChaCha20, is suggested for increasing ciphertext security. Our novel proposal was compared with other random noisy alternatives such as random noisy DES, random noisy 3DES, random noisy AES-256, and random noisy Blowfish. The obtained results were dynamic ciphertext outputs. These schemes are limited to the ASCII table values. In conclusion, the suggested alternatives presented here may be difficult for cybercriminals to decrypt.

KEYWORDS: applications of AI; cryptography; dynamic encryption methods; noisy injection strategies.

RESUMEN

El problema de robo digital de datos en las organizaciones está recibiendo gran atención porque puede ocasionar pérdidas financieras. Este problema se puede amortiguar usando métodos de cifrado dinámico. Existen alternativas seguras para el cifrado de datos, tales como AES (Advanced Encryption Standard) y RSA (Rivest-Shamir-Adleman). Sin embargo, es sabido que dichos algoritmos se encuentran amenazados por la llegada de la computación cuántica. Por lo tanto, el objetivo de esta investigación es recomendar alternativas para encriptado dinámico con inyección de ruido, usando inteligencia artificial (IA), porque ello puede confundir a los ciberdelincuentes. Se estudian aspectos relacionados y aunque no se utiliza computación cuántica, se proponen algunas medidas. El diseño de la metodología consiste en la adaptación del algoritmo ChaCha20, combinado con el método random Caesar II (fusión que ha sido denominada: random noisy ChaCha20), con el propósito de incrementar la seguridad de los textos cifrados. Este nuevo esquema es comparado con otras alternativas aleatorias ruidosas, tales como random noisy DES, random noisy 3DES, random noisy AES-256 y random noisy Blowfish, obteniendo como resultado textos cifrados dinámicos, aunque limitados por valores de la tabla ASCII. En conclusión, las nuevas propuestas podrían ser difícil descifrar para los cibercriminales.

PALABRAS CLAVE: aplicaciones de IA; criptografía; cifrado de datos dinámico; cifrado con inyección de ruido.

Corresponding author:
NAME: Edgar Rangel Lugo
INSTITUTION: Tecnológico Nacional de México/ Instituto Tecnológico de Ciudad Altamirano
ADDRESS: Av. Pungarabato oriente s/n, col. Morelos. C. P. 40660, Ciudad Altamirano, Guerrero, México
E-MAIL: erangel_lugo@hotmail.com

Received: 29 September 2025. Accepted: 8 December 2025.
Published: 31 December 2025.

open access
UACJ

PEER REVIEWED

I. INTRODUCTION

A cybersecurity strategy [1]-[3] is considered inadequate if at least one of the methods is vulnerable to cybercriminal attacks [4]-[8]. This situation can produce the theft of digital data [8]-[9]. When it occurs in practical domains, it may cause significant losses in the finances of organizations [4]-[5], [8], [10]-[11]. Most of these cases [12]-[17] refer to fraudulent telephone calls or phishing, social networking platforms, bank systems, large markets or retail supply chains, electrical energy network business, detecting of fraudulent financial on sector situations, and several cases of e-commerce in organizations [4]-[5].

Several proposals have been developed for combating the theft of digital data. These strategies can be classified into three main approaches: updating cybersecurity strategies [4] on a regular basis, implementing dynamic encryption methods [5], and using noisy injection on ciphertext [5]-[7], [9]-[10]. This scheme has demonstrated potential in certain practical domains.

This research focuses on encryption methods [8] that utilize noisy injection strategies [5]-[7]. By computing mathematical equations or statistics, plaintext (`Si`) is transformed into ciphertext (`Ci`) [8], which can only be accessed by authorized parties [10]-[11]. The `Si` denotes the original input sequence, and the `Ci` is the encrypted output. When encryption method produces distinct results with the same plaintext input is considered dynamic, whereas static encryption schema yields the same result every time [8].

These algorithms can be classified as symmetric, where a single secret key is used, or asymmetric, where a pair of keys (private and public) are employed [8].

The process of translating plaintext into ciphertext is known as data encryption, and the inverse process is called data decryption [8]-[10], [14], [18].

Asymmetric encryption algorithms, including RSA (Rivest-Shamir-Adleman) [3], [5]-[6], [8], [13]-[14], [19]-[27], ECC [5]-[6], [8], [19]-[23], [28]-[31], and ElGamal [13], [19]-[21], require both private and public keys to operate. Recent research [5], [8], [11] have reported dynamic encryption results when these asymmetric alternatives were employed.

In this work, the asymmetric algorithms have not been experimented because it can be considered a future work. Therefore, these schemes are not described in this research.

On the other hand, there exists also various symmetric key cryptography algorithms, such as DES (Data Encryption Standard) [3], [13]-[14], [22]-[23], [27], [32]-[33], TripleDES or 3DES (Triple Data Encryption Standard) [14], [21]-[23], [34], Blowfish [22]-[23], [35]-[38], ChaCha20 [38]-[39], and AES (Advanced Encryption Standard) [13], [21]-[23], [26]-[27], [40]-[41], to name a few. In case of the AES scheme, in this research AES-256 version [38], [40]-[41] has been employed.

According to reference [22], AES is a symmetric block cipher that can operate with varying block sizes and supports key lengths of 128, 192, and 256 bits. However, DES encrypts 64 bits of plaintext into 64 bits of ciphertext, employing substitution and permutation techniques through a series of rounds, and decryption is performed by reversing the process. Besides, the employment of 64 bits, it is considered insufficient for secure environments, making it relatively easy to break. As a result, the 3DES was developed as an enhancement to DES. Blowfish is a symmetric algorithm that uses a variable-length block cipher, supporting key lengths between 32 and 448 bits [22]-[23]. Similarly, Blowfish is commonly implemented with a 64-bit block size.

On the other hand, ChaCha20 [39] is a symmetric algorithm that succeeds Salsa20 and it is built on the ARX cryptographic primitive. ChaCha20's keystream generation algorithm consists of three operations: addition modulo 232, constant distance left bit rotation, and bitwise XOR operation. These operations allow ChaCha20 to achieve high speed and security. ChaCha20 takes a 128-bit or 256-bit key, a nonce, and a 128-bit constant to produce a 512-bit keystream. ChaCha20 introduces a slight modification to its internal state matrix, making it more resistant to certain types of attacks and often faster in software implementations.

The experimentation with the Salsa20 algorithm is beyond the scope of this research and may be pursued in future work.

In this context, some research [5], [8] have revealed that symmetric encryption algorithms can generate static ciphertext outputs. It does not mean that these schemes are vulnerable [5].

However, there exists some standard encryption alternatives that they have been threatened by quantum computing [8], [22]-[23], [42]. In contrast, other researchers [43]-[45] caution that the AES and RSA algorithms are also vulnerable to the emergence of quantum computing [8], [22], [42].

Several aspects of this topic are examined in this paper. However, a quantum computing alternative has not been employed, as other measures are proposed here, which are applied to variants of symmetric algorithms [5]. In this context, the implementation of a noisy injection strategy [5]-[7] is also highly recommended for enhancing encryption security.

Noisy injection involves the addition of characters from ASCII or UTF-8 encoding to a plaintext or ciphertext that exceed the original input message, introducing extraneous elements [8].

One of the hypotheses explored in this paper is that noisy random strategies have the potential to mislead cybercriminals [5], [8]. These noisy injection strategies [5]-[7] often rely on artificial intelligence (AI) [5]-[8], [10]-[11], [46]-[47], given the existence of AI-based cryptography [1]-[4], [8], [10]-[11], [16]-[17], [24], [48]-[50].

References [10], [46]-[47] mention that AI's purpose is to make the machine think [8]. In this regarding, the heuristic methods [5], [8], [10] can help us, because these methodologies consist in a previously defined set of rules for solving a problem. They can be used for implementation of structured models such as decision tree [51]-[53], graphs [53]-[55], to mention few.

Furthermore, random methods in AI [10], [56]-[59] involve selecting numbers randomly, either with or without replacement [8]. These techniques can be applied to intelligent models like genetic algorithms (GAs) [1]-[2], [9]-[10], [16]-[17], [56], [59]-[62], Monte Carlo (MC) algorithms [59], and artificial neural networks [57], [63]-[64], and other applications [65]-[68]. Several authors have explored AI-based cryptography alternatives [1]-[4], [8], [10]-[11], [14], [16]-[17], [24], [27], [48]-[50], [60]-[62]. These schemes include random noisy strategies that were tested [4], [9]-[11] on different platforms, including Microsoft Windows [69] with Python 3 [70], and Android [71] with PyDroid3 [72], using various Python libraries [73]-[75].

For a comprehensive overview of cryptography with AI, see [50]. Reference [48] analyzes the application of GA in the determination of efficient parameters for a specific model of pseudorandom number generators, known as Congruent Linear Generators (CLGs), while [3] focuses on asymmetric and symmetric cryptographic methods.

In [58], a study on pairing functions for AI-driven cryptography was conducted. Subsequently, the same author [16] have published a novel investigation on GAs in cryptography, specifically contributing to the field of e-commerce [16]. Another research [24] highlights a review of side channel attacks and countermeasures on ECC, RSA, and AES cryptosystems. In reference [49], a survey trends in lattice-based cryptographic schemes is presented, including some recent fundamental proposals for the use of lattices in computer security, challenges for their implementation in software and hardware, and emerging needs for their adoption.

Following conventional methodologies, some studies [1], [62] have investigated the application of genetic algorithms in cryptography. Additionally, reference [60] has presented an advanced optimization algorithm tailored for cryptanalysis. On the flip side in [61] reveals that genetic algorithms can successfully break certain simple cryptographic ciphers. In [27], a similar vein is examined the application of genetic operators to symmetric cryptography using GAs. Moreover, [49] introduces a post-quantum lattice-based cryptography implementations.

Noisy injection strategies [4]-[11] encompass multiple approaches that they can be classified into three distinct categories.

Firstly, the use of pseudo-hexadecimal format is considered. In this regard, the 'Noised' random pseudo-hexadecimal GAs methodology has been detailed in [9]-[10]. This scheme, based on a genetic algorithm was introduced as a dynamic encryption solution. However, due to the reported disadvantages of pseudo-hexadecimal GAs, a successor was presented in [10], known as "Noised" random pseudo-hexadecimal (without GAs). In [8], four dynamic alternatives based on the pseudo-hexadecimal scheme were introduced, termed "noisy random pseudo-hexadecimal" strategies. These strategies involve injecting noise into ASCII characters to confuse cybercriminals when a new pseudo-hexadecimal format has been recommended. The application of these schemes is restricted to plaintext.

The second category includes the use of AI-based noisy injection paired with the 1-NN rule, as referenced in

[46]-[47], [65]-[67]. In this regard, the 'Noised' random 1-NN with hexadecimal encoding based on AI has been introduced in [11]. Similarly, the combination of "Noised" random pseudo-hexadecimal format with 1-NN rule was explored in [68]. Both methodologies have revealed that these schemes can increase the safety of digital data with double noisy injection over ciphertext outputs. Even though it will have to sacrifice disk storage space. These strategies were also applied to plaintext. In this study, the pseudo-hexadecimal schemes were not explored, but it may be considered for future research.

In this context, random Caesar II mod 120 [4]-[7], [11] was employed in the third category for noisy injection over plaintext [4]-[5], [11], as well as being applied on ciphertext [5]-[7], which it is generated by standard encryption algorithms. This approach is termed a random noisy strategy [5]-[7].

Some studies [4]-[11] indicate that dynamic encryption methodologies based on random noisy schemes can increase the security of ciphertext outputs by adding noise and redundancy [5], [9]-[10]. In AI-based practical domains [46]-[47], [55]-[56], [63], [65]-[67], the presence of incomplete or noisy patterns [76]-[77] can reduce the systems' global accuracy [66]. Hence, the noisy injection alternative is considered a good indicator because it can mislead cybercriminals.

In contrast to the traditional Caesar algorithm [4], [9], [13]-[14], the random Caesar cipher is distinguished by its use of dynamic encryption with AI, as noted in reference [5], which it emphasizes its use of heuristic methods. While the traditional Caesar cipher relies on a fixed shifting value K, as expressed in equation (1), in references [13]-[14]:

$$\mathtt{Ci = Si + K \bmod 26} \tag{1}$$

In this situation, the random Caesar cipher utilizes varying shifting values (Ki) for each character Si, chosen randomly with replacement.

Random Caesar's mode of operation is determined by the N value in the terms of equation (2):

$$\mathtt{Ci = Si + Ki \bmod 26} \tag{2}$$

Unlike the traditional Caesar cipher, which uses mod 26 and is limited to 26 characters, the random Caesar cipher offers more flexibility. Hence, the mod N in random Caesar method is potentially dynamic. It uses an initial AI-based learning phase [8]-[10] that is recommended for selecting alphabet, but it has been narrowed down to three modes in recent research [4], [6], [9], [11], including the random Caesar I (with mod 9 and mod 255) [6], random Caesar II with mod 95 [4], [9], [11], and random Caesar II with mod 120 [4]-[7]. In these terms, the mod N value determines the size of the encryption alphabet and the maximum value in the Ki vector. For N = 95, the range is 32 to 126, encompassing characters like space and '~'. The N = 120 value, it spans 30 to 150. Any other N value means Ki is between 0 and N. These values are not ordered according to their ASCII code (ordinal) because they have been selected randomly.

The schemes outline the rules for selecting alphabets based on random principles and the use of a heuristic methods to derive the best Ki vector. This situation as well as the use of AI have already been discussed in other studies [4]-[11], [68]. However, it is explained below.

The random Caesar schema's second phase is designed to confuse cybercriminals [4], and involves calculating the final package using the equation (3), in reference [5]:

$$\mathtt{FinalPackage = Ci \ \& \ Ki \ \& \ OrdChr(Ci)} \tag{3}$$

The & operator denotes the concatenation function, and the OrdChr procedure appends the same character of Ci to the end of the package when operating in N = 120 mode. In other cases, OrdChr converts Ci to its ordinal value.

This methodology is specifically designed for plaintext encryption as a dynamic approach [4], whereas studies [5]-[7] have incorporated the random Caesar II mod 120 as a random noisy strategy. Eight random noisy encryption methods are described in [5], including: random noisy DES, random noisy 3DES, random noisy RC4, random noisy Blowfish, random noisy WEP, random noisy AES, random noisy RSA-2048, and random noisy ECIES SECP-256-R1. These proposals combine standard encryption algorithms with the addition of noisy injection through random Caesar II mod 120.

The concept of random noisy GOST was explored in [6], and the random noisy Camellia was highlighted in [7]. Variants of random noisy strategies [4] that incorporate noisy injection have been reported to effectively camouflage ciphertext [5]. Examples include reduced random Caesar [4]-[5] and reduced random mutation [4].

The main objective of these schemes, as outlined in reference [4], consists into camouflage and compress at least ⅓ of the ciphertext. The reduced random Caesar strategy has been applied on plaintext [4] and ciphertext [5]. In case of the reduced random mutation [4] has only been employed using plaintext.

We did not experiment with these reduced random and mutation strategies here, as they may be addressed in future work.

In this work, only four random noisy strategies based on standard symmetric encryption algorithms have been evaluated, including random noisy DES, random noisy 3DES, random noisy AES-256, and random noisy Blowfish. Similarly, a new alternative was developed and it is introduced here as random noisy ChaCha20.

Noisy injection scheme based on random noisy strategies [4]-[7] can be applied to plaintext [4] or ciphertext [5]-[7]. They are recommended due to its strong correlation with dynamic encryption performance. The impact of quantum computing on the security of these schemes has not been studied here [5], [8].

In this context, random noisy strategies have been rarely explored in the literature. As a result, the vulnerabilities of these schemes have not been thoroughly investigated. However, there exist some dynamic encryption approaches that employ asymmetric algorithms [3], [24], [26], [28]-[29], [43]-[44], pseudorandom number generation [40], chaotic maps [18], optical pattern recognition [15], algorithms based on mutation procedures [4], [59], genetic algorithms [1]-[2], [8]-[10], [16], [27], [56], [58], [62], cryptography based on heuristic methods [54], and pseudo-hexadecimal encoding [8]-[10]. These heuristic pseudo-hexadecimal approaches have inspired the development of our proposed random noisy ChaCha20, as its learning phase is derived from these existing schemes but excluding pseudo-hexadecimal encoding.

Given that random noisy strategies have shown promise, this research continues the work of [5]-[7], by examining the ChaCha20 encryption algorithm's potential when it is applied to ciphertext, a gap in existing research that could benefit organizations employing ChaCha20. Besides, this situation opens opportunities to the organizations, regarding the employment of noisy injection based on ChaCha20 scheme. We focus on cybersecurity strategies that utilize noisy random encryption methodologies, specifically exploring the application of noisy injection on ciphertext generated by standard encryption algorithms with the purpose of misleading cybercriminals [8].

The scope of this study was limited to two classes of situations.

First, we compared five standard encryption algorithms (DES, 3DES, AES-256, Blowfish, and ChaCha20) as static encryption schemes for benchmarking against other research findings [5]-[7].

The random noisy ChaCha20 scheme was also implemented as a new method for comparison with other strategies like random noisy DES, random noisy 3DES, random noisy AES-256, and random noisy Blowfish, which were evaluated for their effectiveness in noisy injection over ciphertext.

These strategies involve using random Caesar II mod 120 [4], being applied to ciphertext previously encrypted with a standard algorithm. Both objectives here focus on dynamic encryption as an alternative for random performance.

This study adds to the empirical foundation of AI-based cryptography, particularly since random noisy strategies have been rarely studied.

Recent research [4]-[5], [9]-[11] have noted that the random Caesar II method with mod 255 [4] can produce ciphertext values outside the ASCII table range [5]. However, in practical domains where random noisy strategies were employed [5], these issues have not been encountered.

The random noisy encryption strategies were previously assessed with five-fold cross-validation [5]-[7]. This work expresses their performance in terms of average or global accuracy [46]-[47], [65]-[66]. Here, we report on the experimental results of an extensive investigation into digital data theft cases. This study examined situations where the use of at least one inadequate static encryption method led to vulnerabilities [5].

Initially, the experiments were focused on replacing of the static encryption scheme for recommending the random noisy strategies as dynamic encryption alternative [4]. Moreover, examples of ciphertexts produced by random noisy encryption schemes, they are included in this research.

The assessment of these approaches involved five samples and a modified cross-validation method [4]-[5], [10]. Furthermore, the application of noisy injection on ciphertext output is suggested as it proves to be a reliable indicator of dynamic encryption efficacy.

Besides, a novel approach named here as random noisy ChaCha20 strategy is proposed as a dynamic encryption alternative. The results are also compared against four random noisy schemes based on the DES, 3DES, AES-256, and Blowfish algorithms.

II. METHODOLOGY

The use of static encryption algorithms as a replacement for existing cybersecurity strategies does not ensure the data protection for organizations. Reference [4], dynamic encryption approaches are suggested instead. This study examines the effectiveness of dynamic encryption measures based on random noisy strategies [5], in preventing digital data theft.

This research is considered experimental and exploratory because a novel random noisy ChaCha20 alternative is introduced here for the first time.

This work required the use of hardware, software, and datasets. The experiments were conducted on a personal computer with a 2 GHz CPU, 4 GB of RAM, and 32 GB of free disk space. The software implementation of these encryption methods, including DES, 3DES, AES-256, Blowfish, and ChaCha20, as well as, the novel variants based on random noisy strategies [5] was carried out using Microsoft Windows 10 [69] and Python [70].

To compare our results with those in [5]-[7], we repeated some experiments on a mobile computing device with the same hardware features as the personal computer mentioned earlier, but with Android 9 [71], as the operating system and PyDroid3 [72], for software development. Our experiments showed no significant differences.

Our datasets are training samples (TS) [6]-[7], [46]-[47], [65] with 1000 exemplars, being selected randomly. Each row in the dataset is a pattern with five columns or features.

This data comprises encryption and decryption details, including ciphertext (Ci) represented as a pair (Test1, Test2), as well as encryption time (TC), decryption time (TD), error rate (Error), and class label. Specifically, the pattern is structured as TP = [(Test1, Test2), TC, TD, Error, Label], enabling comparison with other research findings [7]. In Table 1, two ciphertext examples are shown (Test1 and Test2). The Label or plaintext (Si) sequences include the noisy characters, which were represented in Python as follows:

```
Si = ''.join([chr(9619),'W','e','l','c','o',
'm','e',chr(9619),chr(65533)])#'▓Welcome▓?'
```

This Label feature represents the plaintext (Si), simulating a password with added noise characters (i.e. the ordinals 9619 and 65533 values).

Encryption and decryption times were calculated in milliseconds, while the TC, TD, and Error features were represented as double precision values.

The encryption strategy transforms the plaintext sequence into a ciphertext result, structured as a tuple (Test1, Test2), while computing TC, enabling the observation of dynamic encryption results. The ciphertext sequence is decrypted while TD is calculated. Both sequences are stored in TS, with their TD, TC, and Error rates included in a structured pattern format.

This error rate is calculated according to the number of characters that they are incorrect. If the encryption strategy's output ciphertext, it does not match the plaintext (Si), the error rate is determined by the extent of the errors within Si. In this context, if a ciphertext of eight characters corresponds to a plaintext of eight characters and has an error value of 0.5, it indicates that four characters from Test1 and/or Test2 have not been decrypted correctly.

The ciphertext and plaintext are sequences of characters in ASCII or UTF-8 encoding, with a maximum length of 255 characters. Unlike other algorithms, 3DES and Blowfish have limitations in processing block sizes, which limited the experiments with Blowfish to a block size of 13 characters and 3DES to a block size of 22 characters. The selected plaintext sequences in our experiments are intended to simulate passwords. In real-world applications, passwords are typically recommended to have a length between 8 and 16 characters. In our experiments, we were able to simulate passwords of up to 255 characters in length. However, we encountered an issue with the Python 3 libraries used for Blowfish and 3DES encryption, which truncated the plaintext sequences to 13 and 22 characters, respectively. To address this issue, we performed piecewise encryption of the plaintext in blocks of 13 and 22 characters for Blowfish and 3DES, respectively, allowing us to evaluate the algorithms on a more equitable basis.

Another alternative for addressing this disparity, the research employed ciphertexts filled with random hexadecimal values to ensure a more equitable comparison. All this information has been used for converting it in new format based on cross-validation modification [4]-[7]. This updated TS format was employed in each encryption strategy, separately.

For improving the results understanding, in Table 1, the arithmetic mean and its standard deviation are shown, using a plaintext values as above mentioned.

Finally, the plaintexts were processed separately with the encryption algorithms, using the test set (TS) created for each algorithm, including those with random noisy strategies, under equal terms.

TABLE 1
Encryption and Decryption Times with Error Estimates

Encryption method	TC	TD	Error	Test 1	Test 2
DES	***1.14*** (0.5087)	0.19 (0.0187)	***1 (0)***	d535c337be1d94db5b940b75bce124dcda93ffa35791bf07	d535c337be1d94db5b940b75bce124dcda93ffa35791bf07
ChaCha20	***7.03*** (3.6519)	0.39 (0.0118)	0 (0)	e1e3a0ba104d580ff51ba42cf64d7f6a1d	e1e3a0ba104d580ff51ba42cf64d7f6a1d
AES-256	7.31 (3.7093)	0.61 (0.0169)	0 (0)	303030303030303030303030303030319421780fc40c59f14796a598115d8e5139f6f9666f67cb30fcda01f22739eeda	303030303030303030303030303030319421780fc40c59f14796a598115d8e5139f6f9666f67cb30fcda01f22739eeda
Blowfish	7.55 (3.7931)	0.64 (0.0373)	0 (0)	4af747eaabe473251f42200cf8fda7f2	4af747eaabe473251f42200cf8fda7f2
3DES	7.68 (3.9388)	0.50 (0.0184)	0 (0)	d535c337be1d94db5b940b75bce124dc620c51e2380a3d5c	d535c337be1d94db5b940b75bce124dc620c51e2380a3d5c
random noisy DES	***1.63*** (0.4897)	0.58 (0.0389)	***1 (0)***	ÏkÏj5j□[□¡l¡ÍjÍ—d□□h›u>u¿]¿□+□□Q,¿[¿¥l¥□l□‘-‘í‹í□[□ÔrÔº□º›g□□P□ÙwÙ„M„t?tÃaÃ□;□¢=¢tCtb0bV"V´P´ã□ã□*□ã,ãi0iwDwó□ó³M³î□î·„·‰T‰{D□{ŠQŠPP©G©¾X¾Œ\□□UŒ	ÛwÛ¼‡¼ˆUˆ¬w¬ºWº¦s¦uBul5l‘/‘ó□ó½Œ½µQµšašX$XîŠîí‹íšeš¼Z¼šaš_+_i9iœ:œ«t«yDy®L®à}àÀ[ÀŸnŸsAs…Q…ÜxÜ“0“ªFª¨G¨Ë’Ë□j□³M³ë…ë»Z»□[□º…º¿ˆ¿q8q´ƒ´´R´ÐjÐP□P□g□
random noisy ChaCha20	***7.72*** (3.7983)	0.74 (0.0171)	0 (0)	ªEªW&W²M²h5h□□□}□‡%‡¡@¡□_□¨x¨“_□□2–t?t□u□¿□¿¢<¢œ6□□jŸi8iµSµ‹*‹«w«h6hê‡êæ□æ¹ƒ¹□\□½Y½□g□ÌfÌ§q§¶U¶‡V‡ÙuÙ	¡<¡OOèƒèT!T¦E¦c3c±O±□?□¨w¨,ˆ,¦r¦□9□¶□¶¡i¡□^□§A§ÇaÇj5j{J{‘/‘õ”õ¿‹¿šh□□*□ðŠðÊ”Êb.bÕqÕyBy‰#‰,,,´S´À□À□*□
random noisy 3DES	8.45 (4.1892)	0.67 (0.0179)	0 (0)	è„è¨s¨,…,□h□ö“ö¢o¢‡T□□`□□4–¿Z¿¥t¥«G«µ|µZ&ZÎjÎ÷•÷a,aîŒî„K□□S‡tDtÏmÏ½†½¶□¶¦D¦î‹î³N³c2c{I{□l□Ä`Ä¹V¹ŠTŠ§u§S#SªGª·,·§v§ÎiÎ`.`Á□Á]%]Å•Å»Z»‹X‹ÔpÔŠU□□8›	ä□ä□Z□o<o¾‰¾□:□Æ“Æi6ixAxÇeÇ¢=¢†U†ÍiÍ°w°□Z□ò□ò»Y»‹V‹ÖtÖ|C|d0dj:j¯M¯h1h¹„¹±O±³P³Å`ÅwFwX&XÈ”ÈÏkÏó□ói3i[)[„T„å,åX#XŒ[Œù”ù¹‡¹Â□ÂWW”d”ò‘òŠWŠ±M±TTÉfÉ
random noisy AES-256	8.46 (3.9351)	1.00 (0.0119)	0 (0)	yFy½□½j7jj:j¹†¹†V†m«{«˜e˜m=mb/bqAq□M□³ƒ³c0c□P□b/b…U…]*]¾□¾Ç”ÇxHxº‡ºˆX□□`“d4dÅ’Å„T□□f™\,\p=pƒR□□N□□M□□S…q@qUU›c›f6f£=£—4—RR\,\ÕrÕyDyzAz‹%□□S„¡m¡·□·µ|µ¤n¤¢A¢²}²„K„e-e•d•X'Xu@uÏkÏ†N□□"‡_*_□_□zGzo6o¸R¸W!Wõ□õË’Ë¢□l¢†P□□b˜¢<¢¯y¯a*a±N±—5—xEx□_□□)□□$‡ñ□ñß~ßX(X‰X‰¹S¹©w©¿□¿Ÿ"Yq>qj1jœ7œÛvÛÄ`ÄÉhÉ	Z'Z|L|QQ□Q□¡n¡ƒS□□^‘¹‰¹s@s]-]T!TÅ•Å`-`§w§ŠWŠ¬|¬‹X‹³ƒ³Æ“Æ±□±Y&Yb2b±~±…U□□R□□R,\)\°□°œiœ`0`£p£^-^‹R‹h4hµƒµÄ“ÄÃŒÃb*b§w§ÜvÜÂ_Â´□´…U□□0□□L□XXµOµ}L}ŸkŸ¼…¼†M□□_•³R³k6k□G□□bš¦u¦—f□□gœçƒç¾†¾¡<¡k6kƒRƒ½Š½Ç□Ç¤>¤Å□ÅÓmÓ£j£□h□□g□□J□□9Ÿ{E{i2i›8›>□k□yIy□'□ÙvÙ´P´Š)Š^.^ŠYŠ´N´ˆVˆÂ□Â¨q¨T!Tšaš§B§ú•ú°L°ð□ð
random noisy Blowfish	7.85 (3.7619)	0.94 (0.0349)	0 (0)	wCwð□ððŠð□H□”`”]&]•0•ï□ï¨G¨Œ*Œô□ôuAu±z±vCvT"T·,·—f—ó□óšfšPP¤r¤°□°µ…µ™6™¶P□¶zBzš4šô□ôíŒí†O†®H®□k□	±}±ÉhÉê„êzCzT□TË”Ë‘,‘ÂaÂ‘0‘¬J¬…□□Oƒ¹,¹mÅ“ÅW"Wk:kã}ã□Z□{I{»‰»rBr¶†¶ÜyÜ¦@¦k3ká{á¼X¼‘0□□_–ÖpÖU#U
Average	**6.48** (2.5856)	**0.63** (0.2279)	0.2 (0.4000)	- - - -	- - - -

IMPLEMENTING STANDARD SYMMETRIC ENCRYPTION ALGORITHMS

Experiments began by using standard symmetric encryption algorithms, such as DES, 3DES, AES-256, Blowfish, and ChaCha20, as our primary strategy. They were experimented as static encryption schemes being applied on plaintext for results comparison with other research [5]-[7].

These experiments were run on five training samples with the above-mentioned details, and each encryption algorithm, it was assessed separately using a modified cross-validation method [4]-[5]. These standard encryption algorithms were implemented using Python language [70]. For this it was necessary to install some package or libraries such as cryptography [73], pycryptodome [74]-[75], [76], and pycrypto/pycryptor. Therefore, it needs to be imported into the source code and the Si values must be initialized as follows:

```
from cryptography.hazmat.primitives import
padding

from cryptography.hazmat.primitives.ciphers
import Cipher, algorithms, modes

from cryptography.hazmat.backends import
default_backend, from Crypto.Cipher import
DES

Si = ''.join( [ chr(9619), 'W', 'e', 'l', 'c',
'o', 'm', 'e', chr(9619), chr(65533) ] )
```

Given that Si was saved, we can proceed with the analysis. The ciphertext generated using the DES algorithm can be obtained through the following operation:

```
Ci=((A(Key.encode(),Mode)).encrypt(plain
text)).hex()print(Ci)
```

In the case of ciphertext produced by the 3DES and Blowfish algorithms, the computation operation is:

```
Ci=(Ri.update(plaintext)+Ri.finalize()).hex()

print(Ci)
```

In the same way, the ciphertext for the AES-256 encryption alternative, it can be calculated as follows:

```
Ci=(IV.encode()+(Ri.update(plaintext)+
Ri.finalize())).hex()

print(Ci)
```

Below is the ciphertext obtained through the ChaCha20 algorithm:

```
Ci=(Ri.update(plaintext)+Ri.finalize()).hex()

print(Ci)
```

Here, the A component represents the algorithm used, while the encoded Key parameter is the secret key. The IV value is the initialization vector, while that the Mode argument specifies a valid operation mode for the algorithm, and the Nonce refers to the ChaCha20 nonce value that it was employed. The plaintext argument is the encoded Si, and the encrypt() procedure returns a ciphertext object, which is a class component. In this context, N is the maximum byte length of a character sequence. The Ri vector is a partial ciphertext object that may not have padding or may be incomplete. The Qi component is the padding applied. The updated() and finalize() procedures are necessary to complete the encryption process. Finally, the hex() function is used to translate byte values into hexadecimal format. Based on these terms, the computation of valid parameters for the DES algorithm ciphertext generation is as follows:

```
Key = "00000001"

A = DES.new

N = 8

Mode = DES.MODE_ECB

plaintext=Si.encode()+(b"\x00"*(N-len(Si.
encode())%N))
```

The valid parameters for computing ciphertext with the Blowfish algorithm can be obtained through:

```
Key = "00000001"

A = algorithms.Blowfish

N = 16

Mode = modes.ECB()

Ri=Cipher(A(Key.encode()),Mode,default_
backend()).encryptor()

plaintext=(Si+""+str("".join([" " for k
in range(0,int(N-len(Si.encode())))] ))
).encode()
```

To produce ciphertext with 3DES, the valid values can be obtained through:

```
Key = "000000000000000000000001"

A = algorithms.TripleDES

N = 24

Mode = modes.ECB()

Ri=Cipher(A(Key.encode()),Mode,default_
backend()).encryptor()

plaintext=(Si+""+str("".join([" " for k
in range(0,int(N-len(Si.encode()))) ] ))
).encode()
```

The valid values for ciphertext generation using the AES-256 encryption version are as follows:

```
Key = "00000000000000000000000000000001"

A = algorithms.AES ; N = 256

IV= "0000000000000001"

Mode = modes.CBC(IV.encode())

Ri=Cipher(A(Key.encode()),Mode,default_
backend()).encryptor()

Qi = padding.PKCS7(N).padder()

plaintext=Qi.update(Si.encode())+
Qi.finalize()
```

To obtain ciphertext using the ChaCha20 algorithm, the valid values are:

```
Key = "00000000000000000000000000000001"

Nonce = "0000000000000001"

A = algorithms.ChaCha20 ; N = 32

Mode = None

plaintext = Si.encode()

Ri=Cipher(A(Key.encode(),Nonce.encode()),
mode=None,backend=default_backend()).
encryptor()
```

These statements should be added to the source code before calling `Ci`, as necessary.

RANDOM NOISY ENCRYPTION STRATEGIES

A second approach to encryption involves the use of random noisy alternatives [5], for dynamic data encryption. Reference [4] offers a promising way to increase the noise in ciphertext outputs.

These strategies [5] have been applied to ciphertexts generated by standard encryption algorithms, focusing on four specific cases: random noisy DES, random noisy 3DES, random noisy Blowfish, and random noisy AES-256. Additionally, random noisy ChaCha20 is introduced as a new proposal in this study.

The five random noisy strategies were developed in Python [70] and evaluated using a noisy injection application that applies random Caesar II mod 120 to the ciphertext generated by each standard encryption algorithm. The goal was to compare results with existing research [5]-[7].

Each encryption algorithm was evaluated separately on the five TS using an iterative process with five repetitions of cross-validation [4]-[5]. We applied modified cross-validation to calculate the global average and standard deviation for each encryption strategy.

The novel proposals, as described in [5], involve noisy injection into ciphertext, and the procedure for computing random noisy strategies is detailed in [5]-[7] such as follows:

$$\texttt{RandomNoisyi} = \texttt{Char}(\texttt{Ord}(\texttt{StandardEncryptioni}) + \texttt{Ord}(\texttt{Ki})) \mathbin{\&} \texttt{Char}(\texttt{Ki}) \mathbin{\&} \texttt{Char}(\texttt{StandardEncryptioni}) \bmod 120 \tag{4}$$

This calculation was optimized by substituting ciphertext for plaintext, as demonstrated in [4]. The calculation is adjusted to mod 120 since only character types are stored in `RandomNoisyi` (`FinalPackage`). Several random noisy schemes have been presented in previous work [5]. We employed four strategies for obtaining `StandardEncryptioni` ciphertext in this research.

The random noisy ChaCha20 approach was implemented and computed as follows:

$$\texttt{RandomNoisyChaCha20i} = \texttt{Char}(\texttt{Ord}(\texttt{StandardChaCha20Encryptioni}) + \texttt{Ord}(\texttt{Ki})) \mathbin{\&} \texttt{Char}(\texttt{Ki}) \mathbin{\&} \texttt{Char}(\texttt{StandardChaCha20Encryptioni}) \bmod 120 \tag{5}$$

In this context, the + operator is used for the shifting function, and the & operator is used for concatenation. The `Ord` function maps a character or integer to its corresponding ordinal value, while the `Char` function translates its argument into the corresponding ASCII or UTF-8 encoding. The `Ki` vector contains the random (integer) shifting values. The `StandardEncryptioni` parameter represents the ciphertext obtained from a standard encryption algorithm (e.g., DES, 3DES, Blowfish, and AES-256), as described in [5]. The `StandardChaCha20Encryptioni` argument signifies the ciphertext resulting from the ChaCha20 algorithm. These strategies were applied separately. `RandomNoisyi` refers to the `FinalPackage` generated by applying a random noisy strategy, as mentioned in [5]. On the other hand, `RandomNoisyChaCha20i` represents the `FinalPackage`, resulting from the use of random noisy ChaCha20.

The random noisy approaches involve a two-step process: first, the standard encryption algorithm is applied to the plaintext, and then random Caesar II mod 120 is applied to the ciphertext generated in the first step [5]-[7].

As noted in [5], it's essential to distinguish this fusion of techniques from double encryption using different algorithms, which is not the focus of this research. Applying multiple standard encryption algorithms sequentially could introduce vulnerabilities, making it susceptible to decryption through computational methods like iterative attacks.

Based on other research [5], noisy injection should be strategically applied to the ciphertext to avoid revealing the location of the noise and raising unnecessary suspicion. When applied to only part of the ciphertext, cybercriminals would face the intricate challenge of pinpointing the noisy characters' locations, a task that remains formidable even in the field of quantum computing because these random noisy strategies have not been yet studied.

These strategies involve the use of random Caesar II mod 120 [4]-[5], applied to ciphertext previously obtained through a standard encryption algorithm, thereby serving as dynamic encryption alternatives for random performance [4].

Random noisy strategies for information encryption have proven effective in producing dynamic ciphertext, thus improving data security within organizations. Moreover, the random Caesar II methodology (with mod 120) is classified as an AI-based approach due to its use of random and heuristic methods for `Ki` vector selection [5]. As a result, artificial intelligence was used in tandem with the heuristic method to select the `Ki` vector that produces maximum values for the encryption alphabet. The similarity in procedures between the heuristic method and genetic algorithms leads to the consideration of AI application. This situation and the employment based on AI, they have been already discussed by other research [4]-[11], [68]. However, the details are described as below.

Thereby, heuristic method is defined as a validation tool for the selected ASCII characters [5], [8], as outlined in reference [8]. A genetic algorithm (GA) is a random process that encompasses selection, crossover, and mutation phases, followed by an evaluation stage using a wrapper [9]-[10] or fitness function [10], [16], to assess each generation of the GA [1]-[2], [9]-[10], [16], [56], [59]-[62]. In these terms, the random Caesar methodology [4]-[5] relies on the GA selection procedure for selecting alphabets with mod 120 and its corresponding `Ki` vector of shifts. To ensure ASCII compatibility, `Ki` values are restricted to the range of 30 and 150, as maximum.

In this context, the use of artificial intelligence in data encryption based on noisy injection has been explored in previous research [4]-[6], [8]-[10], [11], [68]. These studies explain the use of different alphabets, referred to as modules, with sizes of 9, 95, 105, 120, and 255. Each alphabet corresponds to a specific range of ordinal values or characters in the ASCII table. However, using ordered ranks would make it relatively easy for cyber-criminals to decipher the encrypted data. To address this issue, previous research has proposed several methods for generating optimal alphabets with random values corresponding to the ASCII table [5], [8]-[10]. These processes may involve the use of genetic algorithms, with or without the application of the nearest neighbor rule [11], [68], or even an abbreviated version using heuristic methods [8]-[10], which are all part of pattern recognition and supervised learning. The use of artificial intelligence is therefore justified.

On the other hand, although the standard version of the random Caesar II method exclusively uses a random process with replacement, some studies [5], [8] suggest that the `Ki` vector can be selected using a heuristic method. In this research, a previous learning phase is introduced to generate the encryption/decryption alphabet, using a procedure based on AI, similar to the noisy random pseudo-hexadecimal (by shifting) scheme [8]. However, our ChaCha20-based proposal does not use the pseudo-hexadecimal format. Therefore, the same fitness function as the noisy random pseudo-hexadecimal GAs or pseudo-hexadecimal by shifting version [8]-[10] is used to prevent the alphabet vector from having repeated characters, ensuring the quality of the encryption/decryption. This ensures that the characters in the alphabet are not ordered according to their ASCII ordinal values, making it more difficult for cyber-criminals to decipher. In this context, the operation `Ci = Si + Ki` refers to a substitution-based displacement, rather than a direct operation on the ordinal value. As explained in [13] and [14] with regard to the traditional Caesar algorithm.

On the other hand, the reduced random Caesar strategy [4]-[5] can also employ the GA selection procedure with mod 120, and reduced random mutation [4], it uses the first and third stages of the GA model (i.e., selection and mutation). For both reduced random schemes, the `Ki` shifting range is constrained between 0 and 105 ordinal values to stay within the ASCII table limits.

We excluded reduced random Caesar and reduced random mutation from this study because they may be worth examining in future works.

Regarding result evaluation, a modified cross-validation method [4]-[7] is proposed to internally bias the discrimination process, building on previous discussions. This information can help organizations consider noisy injection as a viable security measure.

In Table 1, the encryption (`TC`) and decryption (`TD`) times (in milliseconds) and estimated errors are presented, with some results rounded for consistency with [5], [7]. The `TC` and `TD` columns display average encryption and decryption times, with standard deviations in parentheses. Two ciphertext tests for each strategy demonstrate the potential for different results, even with the same plaintext, as above mentioned.

III. RESULTS AND DISCUSSION

The experiments were conducted using `TS`, and the estimated error, `TC`, and `TD` were computed individually for each encryption strategy, as previously described. A five-fold modified cross-validation [4] was applied to each TS, enabling a direct comparison with the results of other research [5]-[7].

The traditional cross-validation methodology [46]-[47], [56], [59], [63], [65]-[67] typically involves dividing the TS into five subsamples of roughly equal size (around 20% each), with one subsample serving as the test set (MC) for model evaluation [7]. The four subsamples not used for testing (approximately 80% of TS) because they are combined and used for model training. The trained model is then tested on the MC, which serves as new data for evaluation. This process is repeated five times to derive the standard deviation and the average or global accuracy [66]. In this research, the encryption algorithm's evaluation does not necessitate a training model with TS or evaluation with MC, rendering the traditional procedure inapplicable to data encryption or decryption. Consequently, the training and evaluation tasks were carried out before the ciphertext or `FinalPackage` was generated.

For instance, the standard encryption algorithm is first applied to the plaintext to generate a ciphertext. It is then decrypted and both vectors are saved in the `TS`. In the case of random noisy strategies, the plaintext is encrypted using the standard encryption algorithm for producing a ciphertext. The heuristic method is then applied to emulate a partial phase training, leveraging the GA's random selection stage.

The random Caesar strategy is used for partial training to obtain the `Ki` vector, which is then applied to the ciphertext for noisy injection in the `FinalPackage`. The encrypted sequence is decrypted, and both vectors are saved in the TS.

The cross-validation method [4]-[5] has been modified to adopt a new approach that it does not rely on MC for assessing global accuracy [66]. Such is the case of the process for data encryption/decryption, experienced in this research. This cross-validation modification [4]-[7] consists in omitting the evaluation of the MC patterns. Instead, the error is computed, but only with a part of TS (i.e., only four subsamples are employed).

This operation is repeated five times, being extracted sequentially, approximately 20% of the information (i.e., 20% of TS is omitted).

By excluding part of the TS, this schema can simulate the estimated error in different environments, yielding a more convergent result with an optimistic bias (i.e., the value obtained may be better or equal when applied practically)

[7]. Moreover, it allows evaluating the estimated error and other numeric features of TS. On the basis that, if the decrypted ciphertext, it does not match the input plaintext (class identifier or `Label`), then the error percentage is calculated, according to number of coded characters that they could not be decrypted, to calculate their percentage.

The presented values of the Test 1 and Test 2 columns are approximations of the ciphertext, as they include non-printable characters. These characters may not display correctly on the screen due to their nature. Both tests were copied and pasted exactly from the file created by Python.

The presence of non-printable characters can cause differences in screen representation when formatted in MS Word (.docx) or PDF. However, the underlying ASCII or UTF-8 ordinal values of these characters remain consistent across different document formats. This means that, although the visual representation may change, the actual values do not. This characteristic can actually enhance security, making it more challenging for cybercriminals to interpret the ciphertext.

Similarly, as the traditional cross-validation does, the operation is repeated five times, extracting sequentially, a different subsample in each iteration. With purpose of calculating the average and standard deviation of each attribute or column of numeric type, which in this research, it was applied to the encryption times (`TC`), decryption times (`TD`), and error percentage, globally, without distinguishing the elements by class. Given that encryption ambiguity, it was observed during experimentation, the cross-validation operation was performed without distinguishing elements by class. This aspect may be explored in future work, as it warrants further explanation and analysis. This situation has not had a detrimental effect on the global accuracy of the encryption strategies evaluated.

Therefore, this study focused on experimenting with only two classes of situations.

We started by investigating the performance of the five encryption algorithms (DES, 3DES, AES-256, Blowfish, and ChaCha20), using the static scheme on plaintext. This enabled comparisons with other studies [5]-[7]. This evaluation was focused on random noisy DES, random noisy 3DES, random noisy AES, and random noisy Blowfish, while random noisy ChaCha20 is presented as a novel strategy in this research. Thereby, the five random noisy strategies were experimented separately for noisy injection on ciphertext as dynamic encryption measure.

After processing all the samples for each encryption strategy separately, the global average results were computed using the novel updated cross-validation method [4]-[7], as explained above.

The data is displayed in Table 1, where the standard deviation is also shown in parentheses. Columns TC and TD present the encryption and decryption times, respectively, measured in milliseconds, allowing for comparison with other research [5], [7]. This research terminated the iterative experimental process after producing five repetitions of ciphertext for each encryption approach. The results in Table 1, they include the standard deviation in parentheses, which are based on the average of five sequential experiments evaluated using the updated cross-validation method [5]-[11]. The following parameters were used for the standard encryption methods, as described below.

The parameters for the DES algorithm consisted of a 56-bit key ('00000001'), UTF-8 encoding, ECB mode [20], and hexadecimal output for ciphertext. The DES algorithm was implemented using the pycryptodome package in Python [74]-[75].

The TripleDES (3DES) algorithm employed ECB mode [20] with OpenSSL [21], [34], as the default backend and a 24-bit key ('000000000000000000000001') to generate ciphertext in hexadecimal format. Python's implementation of the 3DES algorithm leveraged the cryptography component from the Cryptography libraries [73].

Regarding the Blowfish parameters, the ECB format [20], with `default_backend()` function based on OpenSSL schema, and the secret key of 16 bits with "00000001" values have been employed. The ciphertext outputs with hexadecimal encoding has been obtained. The implementation was also carried out with Cipher component of Python's cryptography package [73].

For AES-256 experiments, a 256-bit secret key ('00000000000000000000000000000001') and a 128-bit `IV` ('0000000000000001') were employed. The implementation involved CBC mode [20], with OpenSSL [21], [34], and PKCS7 padding [21], [34], with 128 bits, generating ciphertext in hexadecimal format. Python's implementation leveraged the Cipher module from both the cryptography package [74].

In the ChaCha20 implementation, a 32-bit key ('00000000000000000000000000000001') and nonce ('0000000000000001') were used with the `default_backend()` function in 'None' mode. Python's standard settings and the cryptography package [73] were applied with hexadecimal encoding for ciphertext outputs.

On the other hand, most of the experiments were successful. Some tests with the DES algorithm and its random noisy alternative were exceptions because have reported errors. The inclusion of characters outside the ASCII table, like ordinals: 9619 and 65533, in the input might be responsible. Although it's hard to control input data in real applications, the information in Table 1 suggests that most encryption strategies were effective in hiding this issue in the ciphertext.

Reference [5] highlights that random noisy alternatives often struggle with controlling the maximum random value selected within the ASCII table. Notwithstanding the effectiveness of these strategies, an ASCII value can be repurposed as a character in another encoding scheme, like UTF-8.

In our work, the use of mod 120, which keeps values within the ASCII range, meant that these situations did not occur. Apart from specific cases of noisy injection into the plaintext input, as mentioned previously.

The encryption process utilizing ChaCha20 outperformed the 3DES, AES-256, and Blowfish, symmetric algorithms, showing speeds 1.03 to 1.09 times faster. The difference in milliseconds ranged from 0.28 to 0.65 (see Table 1).

The encryption/decryption times are much faster using ChaCha20 algorithm, in this research was observed that this strategy supports plaintext or ciphertext with values greater than 255 characters. Thereby, it can be considered a secure schema if this situation is validated properly.

Similarly, the 3DES alternative has not encountered any errors, but it is limited to supporting a maximum of 22 characters for both plaintext and ciphertext, similar to the Blowfish proposals with 13 maximum. In this research, DES has a character limit of 255 for plaintext or ciphertext. This was handled as mentioned above.

Given the average error rate of 1.0% during data processing, the DES alternative is not considered a reliable option. The presence of an error rate in the decryption process is a characteristic of the DES algorithm. This is a significant concern because DES is often proposed as a fast encryption procedure, but it is vulnerable to errors when incorrect data is entered, such as a character outside the ASCII range in a password. In contrast, our novel proposal, random noisy ChaCha20, does not exhibit this error situation.

In Table 1, several static ciphertext results are reported. These schemes have been obtained by standard encryption algorithms: ChaCha20, DES, 3DES, AES-256, and Blowfish. However, it does not mean, in all cases that they are vulnerable or insecure schemes.

Besides, the best balance was obtained with random Caesar when it was applied to plaintext. Obviously, the improvement of encryption times with application on plaintext of the random Caesar II with mod 120 can be faster than the rest of strategies here evaluated.

Experimental results showed an average encryption time of 0.14 milliseconds with a standard deviation of 0.0108, and an average decryption time of 0.05 milliseconds with a standard deviation of 0.0011. These findings are not included in Table 1, as the study's primary objective is to compare standard encryption algorithms with their noisy counterparts.

However, the ChaCha20 combined with random Caesar II mod 120 (named here random noisy ChaCha20 strategy) has shown to be faster than random noisy 3DES, random noisy AES-256, and random noisy Blowfish, of random noisy proposals here experimented, when they have been applied to ciphertext.

The random noisy ChaCha20 strategy based on ChaCha20 and random Caesar II mod 120 has showed superior speed when applied to ciphertext, regarding to random noisy 3DES, random noisy AES-256, and random noisy Blowfish, methods here tested. In the same vein, the most balanced results are also achieved with DES schemes that incorporate its random noisy strategy, despite their disadvantages.

Concerning the novel random noisy ChaCha20 alternative has shown to be between 1.01 and 1.09 times faster than random noisy 3DES, random noisy AES-256, and random noisy Blowfish, of the random noisy strategies here studied. With a range of 0.13 to 0.74 milliseconds as difference.

A comparison of ChaCha20 and its random noisy version indicates that the encryption speed difference is not substantial. Traditional ChaCha20 is 1.09 times faster than the random noisy strategy, with a difference of just 0.69 milliseconds (see Table 1). Both ChaCha20 alternatives tested in this research had a plaintext length limit of 255 characters, as discussed above.

The experiments conducted did not encounter any issues of this limitation. In any case, it is considered that this measure alone produces a considerable improvement in the trust of encryption strategy performance.

Random noisy schemes show promise for experimental procedures, but additional factors require examination given the constraints of this study, where all tests were limited to 255 characters in plaintext. Under the same conditions, the 3DES and Blowfish alternatives were employed, as previously discussed.

Each encryption strategy was evaluated based on its own training sample, being designed independently. However, the random Caesar schemes have been shown to increase data security in organizations [4]-[5], [9], [11], and the results of random noisy strategies yield similar positive outcomes. As shown in Table 1, the global average calculation reflects this effect. Hence, the random noisy approaches yield ciphertexts that they are slightly more extensive. The results indicated that DES-generated ciphertext is faster than that of ChaCha20, 3DES, AES-256, and Blowfish.

Nevertheless, the security implications of using DES are significant, as its ciphertext may be susceptible to decryption. The experiments further revealed an average error rate of 1.0% during the encryption and decryption processes. The application of random noisy strategies to standard encryption algorithms resulted in dynamic ciphertext outputs in all cases.

In our research, we aim to highlight that DES, despite its reported fast encryption schemes, exhibited errors in decryption in our experiments. This situation is not considered a good indicator. The error is attributed to the input data, rather than the encryption process itself. In practical domains, advanced users often incorporate non-standard characters into their passwords, such as non-printable symbols or special characters. The DES algorithm performs well when evaluated using printable characters, but encounters issues when processing non-standard characters. Specifically, when we input the characters '▓' (ordinal 9619) and '?' (ordinal 65533) into the password or plaintext, DES is the only algorithm that fails, whereas the other strategies do not exhibit this issue. Therefore, our proposal, based on random noisy ChaCha20, aims to improve upon these schemes.

In these terms, character errors with ordinal values outside the ASCII table range are not a result of the encryption/decryption process. Rather, these errors occur when a user enters a plaintext, simulating a password, that includes characters that are not part of the ASCII table. For example, the characters '▓' y '?' with ordinal values 9619 and 65533, respectively, they are not valid ASCII characters. While this situation could be addressed by working with binary data, it is considered outside the scope of the current study, which focuses on the injection of noise into plaintext and ciphertext. We are currently not working with files in different formats.

Despite that the processing time for `FinalPackage` ciphertext was greater than that of the standard algorithms, as evident in Table 1's Test 1 and Test 2. ChaCha20 proposals show faster execution times relative to the 3DES, AES-256, and Blowfish strategies. The random noisy schemes, nonetheless, consistently yield dynamic ciphertext outputs. Cybercriminals would encounter significant obstacles in decrypting data, as they would need to determine each random `Ki` shifting value in advance, which it has been previously hidden.

Moreover, when the novel partial noisy injection schema is used which it is presented in [5], named as `PartialNoisy` proposal. The decryption process becomes in very confused task when this additional variant is employed. These tasks of discovering data might be very hard because it has not been yet studied, including quantum computing. No experiments were carried out with `PartialNoisy`. It is considered a future investigation.

In this context, repeated application of these random noisy strategies can obtain a dynamic and better results in comparison with the traditional static encryption algorithms.

This noisy injection alternative can increase the security degree of the ciphertext or plaintext. Besides, this situation might warn us against future quantum computing attacks [8], [22]-[23], [42], improving the digital data security of the organizations, as mentioned above.

Additionally, the resilience of these random noisy alternatives to various cyberattacks remains unevaluated, leaving potential vulnerabilities unknown. In previous research [4]-[5] have recommend utilizing downsized ciphertext with reduced random or mutation approaches [4] as reliable indicators.

These approaches allow for efficient encryption and short ciphertexts, while ensuring data security is not compromised, as the partial ciphertext is secured through `Ki` shifting before being stored.

Therefore, these alternatives can present a low risk for digital data theft by inserting a larger proportion of noise into the ciphertext. The use of reduced random and mutation schemes could be promising. However, this work does not cover these options because they are potential topics for future works.

Notwithstanding the difficulties, the study's goals and hypotheses were fulfilled as planned. By utilizing noisy injection, the random noisy ChaCha20 offers a novel approach to dynamic encryption, yielding ciphertext outputs that they are unique each time. This approach can lead to diverse results, even with the same plaintext and parameters, potentially misleading and hindering cybercriminals' efforts. This information can be corroborated in Table 1.

Our analysis of standard encryption algorithms versus random noisy strategies indicates that noisy injection can be a safe and effective alternative for organizations. It includes the novel random noisy ChaCha20 strategy, particularly in those environments, which have adopted the use of traditional ChaCha20.

The random noisy scheme is recommended to enhance digital data security in this type of cryptosystem. It is considered a safe measure for organizations because the simple fact of having this novel alternative based on noisy injection. It opens a wide range of opportunities for organizations regarding its use because it guarantees improvement in the security of digital data.

A potential area for further research is modifying the dynamic encryption methodology presented in this study to incorporate strategies such as reduced noisy schemes [5] and reduced random mutation [4]. Applying these strategies to the camouflaging ciphertext has shown a reduction of up to 33% in ciphertext size in `FinalPackage` compared to random noisy approaches. Another strategy that could be examined is the application of different methods for noisy injection. Particularly, approaches that combine the simultaneous random noisy methodology with AI based on the nearest neighbor rule [8], [10]-[11], [46]-[47], [65], [67], and pseudo-hexadecimal encoding [8]-[10], as mentioned previously. They are worth exploring, presenting numerous avenues for further investigation in future studies.

IV. CONCLUSIONS

The updating of cybersecurity strategy periodically such as encryption methods, it is one of the factors with a great influence for safety digital data in organizations. However, it does not guarantee their digital data security. Recent research highlight the existence of multiple methodologies examining the issues related to encryption vulnerabilities. A strategy that is too well-known can become compromised and ineffective. In previous studies have proposed various dynamic encryption alternatives to address the issue of digital data theft, as mentioned above.

This paper presents a novel modification of these methodologies. It is based on a fusion of techniques with a standard encryption algorithm combined with random Caesar methodology, for use in real applications of the organizations.

A new dynamical encryption proposal, known as the random noisy ChaCha20 strategy is presented in this paper. Additionally, a comparison of dynamic encryption alternatives based on five random noisy strategies were conducted.

These methods use artificial intelligence to inject noise into ciphertext, relying on random and heuristic approaches as outlined above. Given its capabilities, it is well-suited for deployment in actual organizational environments. Due to that the methodology based on noisy injection offers an important contribution to amend deficiencies, which were produced by inadequate standard encryption strategies. Thereby, it can increase its usefulness.

Experimental findings with dynamic random noisy encryption alternatives have confirmed their capacity for handling cyberattacks and data security issues with high levels of assurance. Notably, these random noisy strategies consistently yield dynamic and generalized results that surpass those achieved with standard encryption algorithms (see Table 1).

We aim to explore this issue in more depth through additional research. One technique we will be investigating involves implementing measures for reducing the size of ciphertext generated by random noisy strategies. One approach could be to utilize reduced random schemes or reduced mutation strategies, as mentioned above, which facilitate the concealment of ciphertext operations.

Another option worth considering is the utilization of multiple methods for noisy injection. Particularly, the techniques that merge simultaneous random noise with nearest neighbor-based AI and pseudo-hexadecimal encoding, as outlined above. Naturally, this opens up a wide range of possibilities that we plan to explore in future work.

REFERENCES

[1] B. Delman, “Genetic Algorithms in Cryptography,” M.S. thesis, Dept. of Computer Engineering, Rochester Institute of Technology, Rochester, New York, 2004. [Online.] Available: https://repository.rit.edu/theses/5456/

[2] S. Kalsi, H. Kaur, and V. Chang, “DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation,” *J Med Syst*, vol. 42, no. 17, Dec. 2018, doi: 10.1007/s10916-017-0851-z.

[3] J. C. Mendoza, “Demostración de Cifrado Simétrico y Asimétrico,” *Ingenius*, no. 3, pp. 46-53, 2008.

[4] E. Rangel-Lugo and K. U. Rangel-Ríos, “Novel Random Encryption Methods Based On Mutation Strategies Of Artificial Intelligence,” *Sci. Pract. Cyber Secur. J.*, vol. 8, no. 3, pp. 84-91, Sep. 2024.

[5] E. Rangel-Lugo, K. U. Rangel-Ríos, and L. González-Vidales, “Dynamic Encryption Methods Based On Noisy Injection And Camouflaging Ciphertext Strategies With Artificial Intelligence,” *Sci. Pract. Cyber Secur. J.*, vol. 9, no. 1, pp. 82-104, Mar. 2025.

[6] E. Rangel, K. U. Rangel, and L. González, “Inyección de Ruido para Encriptado de Datos Dinámico con Inteligencia Artificial. Caso de Estudio: Algoritmo GOST R 34.12-2015,” *Rev. Electron. Divulg. Investig.*, vol. 29, pp. 11-36, Jun. 2025.

[7] E. Rangel and K. U. Rangel, “Mejorando la seguridad del algoritmo Camellia, mediante la inyección de ruido sobre textos cifrados utilizando procesos basados en inteligencia artificial,” *INTELETICA*, vol. 2, no. 4, pp. 75–101, Sep. 2025, Accessed: Sep. 3, 2025. [Online]. Available: https://inteletica.iberamia.org/index.php/journal/article/view/45

[8] E. Rangel, K. U. Rangel, L. González, A. Ortiz, and C. A. Rodríguez, “Four Dynamic Encryption Alternatives With Artificial Intelligence Based On Pseudo-Hexadecimal Noisy Injection Schema For Handling The Theft Of Digital Data Problem,” *Sci. Pract. Cyber Secur. J.*, vol. 9, no. 3, pp. 59-77, Jun. 2025. [Online]. Available: https://journal.scsa.ge/papers/four-dynamic-encryption-alternatives-with-artificial-intelligence-based-on-pseudo-hexadecimal-noisy-injection-schema-for-handling-the-theft-of-digital-data-problem/

[9] E. Rangel, K. U. Rangel, J. Medrano, C. A. Bernal, and L. González. (Nov. 2023). Algoritmo Genético para Cifrado de Datos, Basado en un Nuevo Concepto Pseudo-Hexadecimal con Inteligencia Artificial. Presented at Sexto (VI) Congreso Nacional de Investigación en Ciencia e Innovación de Tecnologías Productivas, Ciudad Altamirano, Guerrero, México. [Online]. Available: https://www.cdaltamirano.tecnm.mx/index.php/17-vi-congreso-nacional-de-investigacion-en-ciencia-e-innovacion-de-tecnologias-productivas/140-tecnm-40

[10] E. Rangel, K. U. Rangel, and L. González, “Cifrado de Datos Dinámico con Inteligencia Artificial, Utilizando el Nuevo Formato Pseudo-Hexadecimal,” *Rev. Electron. Divulg. Investig.*, vol. 28, pp. 46-73, Dec., 2024. [Online]. Available: https://sabes.edu.mx/revista-electronica/27/#

[11] E. Rangel Lugo and K. U. Rangel Ríos, “La regla del vecino más cercano como alternativa para inyectar

ruido a mensajes encriptados por el algoritmo: Noised Random Hexadecimal", *INTELETICA*, vol. 1, no. 2, pp. 1–15, Dec. 2024, Accessed: Mar. 23, 2025. [Online]. Available: https://inteletica.iberamia.org/index.php/journal/article/view/16

[12] D. Álvarez, "Algunos Aspectos Jurídicos del Cifrado de Comunicaciones," *Derecho PUCP*, no. 83, pp. 241-264, 2019, doi: 10.18800/derechopucp.201902.008.

[13] F. Barranco and C. Galindo, "Criptografía básica y algunas aplicaciones." repositori.uji.es. https://repositori.uji.es/items/35da2f29-ee4a-4dbc-a82f-c450a81cf9be (accessed Apr. 13, 2025).

[14] S. Gómez, J. D. Arias, and D. Agudelo, "Cripto-Análisis sobre Métodos Clásicos de Cifrado," *Scientia et Technica*, vol. XVII, no. 50, pp. 97-102, Apr. 2012.

[15] B. Javidi and J. L. Horner, "Optical Pattern Recognition for Validation and Security Verification," *Opt. Eng.*, vol. 33, no. 6, pp. 1752-1756, Jun. 1994, doi: 10.1117/12.170736.

[16] B. Reddaiah, "A Study on Genetic Algorithms for Cryptography," *Int. J. Comput. Appl.*, vol. 177, no. 28, pp. 1-4, Dec., 2019, doi: 10.5120/ijca2019919509.

[17] C. Sebas. "¿Qué son los Algoritmos Genéticos en las Inteligencias Artificiales?" aprendeinformaticas.com. Accessed: Mar. 23, 2024. [Online]. Available: https://aprendeinformaticas.com/algoritmos-geneticos-que-es/

[18] S. Paul, P. Dasgupta, P. K. Naskar, and A. Chaudhuri, "Secured image encryption scheme based on DNA encoding and chaotic map", *Rev. Comput. Eng. Stud.*, vol. 4, no. 2, pp. 70-75, Jun. 2017. doi: 10.18280/rces.040206.

[19] R. Oppliger, *Contemporary cryptography*, 1st ed. Boston/London: Artech House Computer Security Library, 2005.

[20] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. Chapman and Hall Book/CRC Press, 2019.

[21] H. C. A. Van-Tilborg, Ed., *Encyclopedia Of Cryptography And Security*, 1st ed. Springer, 2025, pp. 114-115, 201-202, doi: 10.1007/0-387-23483-7.

[22] L. Baklaga, "Leading The Way In Quantum-Resistant Cryptography For Everyday Safety", *Sci. Pract. Cyber Secur. J.*, vol. 8, no. 3, pp 65-73, 2024. Accessed: Mar. 23, 2025. [Online]. Available: https://journal.scsa.ge/papers/leading-the-way-in-quantum-resistant-cryptography-for-everyday-safety/

[23] R. Bavdekar, C. Eashan-Jayant, A. Ankit, and K. Tiwari, "Post Quantum Cryptography: A Review of Techniques, Challenges, and Standardizations," presented at 2023 International Conference on Information Networking (ICOIN), 2023.

[24] L. A. Tawalbeh, H. Houssain, and T. F. Al-Somani, "Review of Side Channel Attacks and Countermeasures on ECC, RSA, and AES Cryptosystems," *J. Internet Technol. and Secur. Trans.*, vol. 5, nos. 3/4, Sep./Dec. 2016.

[25] D. Luciano and G. Prichett, "Cryptology: From Caesar Ciphers To Public-key Cryptosystems," *Col. Math. J.*, vol. 18, no. 1, pp. 2-17, 1987, doi: 10.1080/07468342.1987.11973000

[26] S. J. Saydahd, R. K. Muhammed, S. A. Hassan, and A. M. Aladdin, "A Comparative Performance Evaluation of Hybrid Encryption Techniques Using ECC, RSA, AES, and ChaCha20 for Secure Data Transmission," *IJOIR*, vol. 12, no. 2, pp. 157–172, Dec. 2025, doi: 10.53523/ijoirVol12I2ID598.

[27] J. Rodríguez, "Operadores Genéticos Aplicados a la Criptografía Simétrica," proyecto de grado, Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, Bogotá, Colombia, 2020. Available: https://repository.udistrital.edu.co/handle/11349/28192

[28] K. Lakshmi Harsha Vardhan and V. Jain, "Enhanced Secure File Transfer: A Comparative Analysis of Elliptic Curve Cryptography vs. RSA," *2025 International Conference on Advanced Computing Technologies (ICoACT)*, Sivalasi, India, 2025, pp. 1-6, doi: 10.1109/ICoACT63339.2025.11005106.

[29] D. Hankerson, J. López, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," in *Cryptographic Hardware and Embedded Systems — CHES 2000. CHES 2000. Lecture Notes in Computer Science*, vol. 1965, Ç. K. Koç and C. Paar, Eds., Berlin, 2000, doi: 10.1007/3-540-44499-8_1.

[30] P .L. Montgomery, "Speeding up the Pollard rho method," *Math. Comp*., vol. 48, no. 177, pp. 453-456, 1987.

[31] NIST, "Recommended methods for key establishment using public key cryptography," NIST Special Publication 800-56A Revision 2, 2013. Accessed: Mar. 23, 2025. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-56ar2.pdf

[32] H. W. Dhany, F. Izhari, H. Fahmi, M. Tulus, and M. Sutarman, "Encryption and Decryption using Password Based Encryption, MD5, and DES," in *Proceedings of the International Conference on Public Policy, Social Computing and Development 2017 (ICOPOSDev 2017)*, 2018, doi: 10.2991/icoposdev-17.2018.57.

[33] M. I. Bhat and K. J. Giri, "Impact of Computational Power on Cryptography," in Multi*media Security. Algorithms for Intelligent Systems*, K. J. Giri, S. A. Parah, R. Bashir, and K. Muhammad, Eds. Singapore: Springer, 2021, doi: 10.1007/978-981-15-8711-5_4.

[34] H. C. A. van Tilborg and S. Jajodia, *Encyclopedia Of Cryptography and Security*. New York: Springer, 2011, doi: 10.1007/978-1-4419-5906-5.

[35] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Fast Software Encryption. FSE 1993. Lecture Notes in Computer Science*, vol. 809, R. Anderson, Ed., 1994, doi: 10.1007/3-540-58108-1_24.

[36] B. Schneier, *Secrets and lies: Digital security in a networked world*. Wiley, 2000.

[37] E. A. AL-Maqtari and E. A. AL-Maqtari, "Performance Evaluation for AES, Blowfish, DES, and 3DES Cryptography Algorithms," *PUIRP*, vol. 2, no. 5, pp. 86-95, Oct. 2024, doi: 10.5281/zenodo.13974870.

[38] R. K. Muhammed *et al.*, "Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption", *KJAR*, vol. 9, no. 1, pp. 52–65, May. 2024, doi: 10.24017/science.2024.1.5.

[39] H. K. Garai and S. Dey, "A multi-step key recovery attack on reduced round Salsa and ChaCha," *Cryptologia*, vol. 49, no. 3, pp. 252–267, Jun. 3, 2024, doi: 10.1080/01611194.2024.2342918.

[40] A. Saini, A. Tsokanos, and R. Kirner, "CryptoQNRG: a new framework for evaluation of cryptographic strength in quantum and pseudorandom number generation for key-scheduling algorithms," *J. Supercomput.*, vol. 79, pp. 12219–12237, Jul. 2023, doi: 10.1007/s11227-023-05115-4.

[41] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002, doi: 10.1007/978-3-662-04722-4.

[42] M. Iavich, T. Kuchukhidze, and A. Gagnidze, "Post-quantum Digital Signature Using Verkle Trees And Lattices," *Sci. Pract. Cyber Secur. J.*, vol. 8, no. 3, pp. 35-52, 2024.

[43] P. Fuegner. "Are RSA and AES Both at Risk From the Quantum Threat?" QuSecure.com. Accessed: Mar. 8, 2025. [Online]. Available: https://www.qusecure.com/are-rsa-and-aes-both-at-risk-from-the-quantum-threat/#:~:text=The emergence of quantum computers,efficiently factoring large prime numbers

[44] M. Sharma, V. Choudhary, R. S. Bhatia, S. Malik, A. Raina, and H. Khandelwal, "Leveraging the power of quantum computing for breaking RSA encryption," *Cyber-Physical Systems*, vol. 7, no. 2, pp. 73–92, 2021, doi: 10.1080/23335777.2020.1811384.

[45] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis," *Int. J. Emerging Technol. Adv. Eng.*, vol. 1, no. 2, pp. 6-12, Jan. 2011.

[46] E. Rangel, "Vecinos Envolventes para Variantes de la Regla del Vecino más Cercano," tesis de maestría, Instituto Tecnológico de Toluca, Metepec, México, 2002.

[47] E. Rangel, "La Regla de los *k* Vecinos más Cercanos (*k*-NN) Basada en Distancia de Manhattan (City-Block) para Mejorar la Clasificación de Patrones," in *Quinto (V) Congr. Nal. de Invest. en Ciencia e Innov. de Tecnol. Productivas*, Cd. Altamirano, Gro., México, Nov. 2022. [Online]. Available: http://erangel.coolpage.biz/pappers/edgarrangel2022.pdf

[48] J. C. Hernández, "Técnicas de inteligencia artificial en criptología," tesis doctoral, Universidad Carlos III de Madrid, 2002. [Online]. Available: https://dialnet.unirioja.es/servlet/tesis?codigo=194087

[49] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-Quantum Lattice-Based Cryptography Implementations: A Survey," *ACM Comput. Surv.*, vol. 51, no. 6, article 129, pp. 1-41, 2019, doi: 10.1145/3292548

[50] Ö. Suçeken and O. Özkaraca, "Cryptography with Artificial Intelligence: An Overview," in *Futuristic Computational Systems and Advanced Engineering for the Society*, J. Hemanth, U. Kose, N. Ibadov, I. S. Uncu, and H. Armagan, Eds. Springer, 2025, doi: 10.1007/978-3-031-94600-4_13.

[51] T. M. Mitchell, *Machine learning*, 2nd Ed. McGraw-Hill, 2020.

[52] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1993.

[53] S. J. Russell and P. Norvig, *Inteligencia artificial: Un enfoque moderno*, 4th Ed. Pearson, 2020.

[54] R. Morelli, R. Walde, and W. Servos, "A study of heuristic approaches for breaking short cryptograms," *Int. J. Artif. Intell. Tools*, vol. 13, no. 01, pp. 45-64, 2004, doi: 10.1142/S0218213004001417.

[55] J. S. Sánchez, F. Pla, and F. J. Ferri, "Prototype selection for the nearest neighbor rule through proximity graphs," *Pattern Recognition Letters*, vol.18, no. 6, pp. 507-513, Jun. 1997, doi: 10.1016/S0167-8655(97)00035-4.

[56] L. I. Kuncheva and L. C. Jain, "Nearest Neighbor Classifier: Simultaneous editing and feature selection," *Pattern Recognition Letters*, vol. 20, no. 11–13, pp. 1149–1156, Nov. 1999, doi: 10.1016/S0167-8655(99)00082-3.

[57] K. P. Murphy, *Probabilistic machine learning: An introduction*. MIT Press, 2022.

[58] B. Reddaiah, "A Study on Pairing Functions for Cryptography," IJCA (0975-8887), vol. 149, no. 10, pp. 4-7, Sep. 2016.

[59] D. B. Skalak, "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms," in *Proc. of the 11th Int. Conf.*, Jul. 10–13, 1994, pp. 293-301, doi: 10.1016/B978-1-55860-335-6.50043-X.

[60] A. Clark, "Modern optimisation algorithms for cryptanalysis," *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*, Brisbane, QLD, Australia, 1994, pp. 258-262, doi: 10.1109/ANZIIS.1994.396969.

[61] W. Griindlingh and J. H. Van-Vuuren, "Using Genetic Algorithms to Break a Simple Cryptographic Cipher," submitted 2002, accessed: Mar. 31, 2003, unpublished.

[62] R. A. J. Matthews, "The use of genetic algorithms in cryptanalysis," *Cryptologia*, vol. 17, no. 2, pp. 187-201, Jun. 1993, doi: 10.1080/0161-119391867863.

[63] L. Bruzzone and S. B. Serpico, "Classification of Imbalanced remote-sensing data by neural networks," *Pattern Recognition Letters*, vol. 18, no. 11-13, pp. 1323-1328, Nov. 1997, doi: 10.1016/S0167-8655(97)00109-8.

[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2021.

[65] R. Barandela, J. S. Sánchez, V. García, and E. Rangel, "Strategies for Learning in Class Imbalance Problems," *Pattern Recognition*, vol. 36, no. 3, pp. 849-851, Mar. 2003, doi: 10.1016/S0031-3203(02)00257-1.

[66] D. Lewis and J. Catlett, "Heterogeneous Uncertainty Sampling for Supervised Learning," *Proc. of the 11th Int. Conf. on Machine Learning, ICML'94*, New Brunswick, New Jersey, Morgan Kaufmann, pp. 148-156, 1994.

[67] T. Cover and P. Hart, "Nearest neighbor pattern classification," in *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[68] E. Rangel and K. U. Rangel, "Novel Pseudo-Hexadecimal Encryption Strategies For Camouflaging Ciphertext Based On Nearest Neighbor With Artificial Intelligence," *IJCOPI*, manuscript in review since 2024, unpublished.

[69] Microsoft. "Descarga de software." Microsoft.com. Accessed: Jun. 1, 2015. [Online]. Available: https://www.microsoft.com/es-mx/software-download

[70] Python. "The Python Network." Python.org. Accessed: Nov. 18, 2024. [Online]. Available: https://www.python.org/downloads/

[71] Google. "Sistema operativo para dispositivos móviles." Android.com. Accessed: Jun. 1, 2025. [Online]. Available: https://www.android.com/intl/es_es/android-12/

[72] Google. "Pydroid 3 versión 7.4_arm64. IDE for Python 3. Lenguaje de programación y compilador." Play. Google.com. Accessed: Jun. 1, 2025. [Online]. Available: https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&pli=1.

[73] Python. "Cryptography 45.0.4." pypi.org. Accessed: Jun. 1, 2025. [Online]. Available: https://pypi.org/project/cryptography/

[74] PyCryptodome. "Crypto.Cipher package. Introduction." pycryptodome.readthedocs.io. Accessed: Mar. 30, 2025. [Online]. Available: https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html

[75] Python, "Pycryptodome 3.21.0." pypi.org. Accessed: Dec. 13, 2024. [Online]. Available: https://www.pycryptodome.org/src/changelog#september-2024

[76] R. Barandela, E. Rangel, J. S. Sánchez, and F. J. Ferri, "Restricted Decontamination for the Imbalanced Training Sample Problem," in *Pattern Recognition, Speech and Image Analysis*, A. Sanfeliu and J. Ruiz-Shulcloper, Eds. Springer-Verlag, 2003, pp. 424-431, doi: 10.1007/978-3-540-24586-5_52.

[77] R. Barandela, J. S. Sánchez, and E. Rangel, "Two Modifications of the Decontamination Methodology," *IASTED*, pp. 391-396, 2003. Accessed: Jun. 1, 2025. [Online]. Available: https://www.actapress.com/PaperInfo.aspx?PaperID=15031&reason=500

ACKNOWLEDGMENTS

This work was supported by Tecnológico Nacional de México (Instituto Tecnológico de Ciudad Altamirano). This research is future work of a last project identified by: 19329.24-P.