Scientific and Practical Cyber Security Journal (SPCSJ) 9(3): 59 – 77 ISSN 2587-4667 Scientific Cyber Security Association (SCSA)

FOUR DYNAMIC ENCRYPTION ALTERNATIVES WITH ARTIFICIAL INTELLIGENCE BASED ON PSEUDO-HEXADECIMAL NOISY INJECTION SCHEMA FOR HANDLING THE THEFT OF DIGITAL DATA PROBLEM

Rangel-Lugo, Edgar[1], Rangel-Ríos, Kevin Uriel[1], González-Vidales, Leonel[1], Ortiz-Mariano, Amparo[1] and Rodríguez-Torres, César Del Ángel[1]
Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Altamirano[1]

ABSTRACT: The theft of digital data problem is receiving growing attention. This situation can produce losses in the organisation's finances, when the used cybersecurity strategy, it is not adequate such as the employment of static encryption methods. There exist several safe alternatives based on standard encryption algorithms such as AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), to mention a few. However, it is known that these algorithms, they have been threatened by quantum computing advent. In some cases, it has been handled with dynamic encryption methodology, which it can create different ciphertext results with the same plaintext input. This schema can help to mistake the cybercriminals. In this paper, several aspects related to this subject, they are here studied although quantum computing is not used in this work, other measures have been proposed. We recommend the replacement of static encryption method by another dynamic alternative. Recent proposals recommend using pseudo-hexadecimal encoding with artificial intelligence. This research is future work of these strategies because it has been observed that these proposals, they can take a long time for processing. This situation might be very good for data protection of the organisations. That's why the importance of our work because that pseudo-hexadecimal encoding with artificial intelligence schema, it has been presented as a good alternative. Regarding the research's aims, they consist in the noisy injection and to avoid that ciphertext can be decrypted by cybercriminals. Four novel proposals based on pseudo-hexadecimal noisy injection strategy that they were named here as noisy random pseudo-hexadecimal standard version, noisy random pseudo-hexadecimal by substitution, noisy random pseudo-hexadecimal by shifting, and noisy random pseudo-hexadecimal version 2, they are here recommended as dynamic encryption alternatives. Finally, the comparison of results with some standard encryption algorithms, it is also introduced.

KEYWORDS*: Dynamic encryption methods, applications of AI, cryptography, noisy injection, pseudo-hexadecimal encoding*

1. INTRODUCTION

The theft of digital data can produce losses in the organisation's finances, if the used cybersecurity strategy, it is not adequate such as the employment of static encryption methods (Rangel et al. 2024). The cybersecurity strategy is a set of procedure or measures that they are designed for safeguarding of threat and cyberattacks in the organizations. These strategies are commonly focused over the security of computing system, computer network, and digital data. Examples of cybersecurity strategies, they can be considered such as risk evaluation, security policies, anti-cyberattack software tools, response incident planning, the employment of firewall antimalware and antivirus, encryption methods, data authentication, updating of system, networking fragmentation, penetration test, backups, to mention a few. In this work, we have only focused over the study of dynamic encryption methods. A cybersecurity strategy is not adequate if at least one of the methods, it is vulnerable to cybercriminal attacks (Rangel and Rangel 2024a). A fundamental definition of cyber crime is any criminal activity involving the use of the internet or cyberspace as a means to carry out the intended dishonest conduct for financial gain or other forms of dishonesty. Cybercriminals target particular computers, the most prevalent cybercrimes today include pharming, phishing, skimming, eavesdropping, and DOS attacks (Drzani 2014; Mamman et al. 2024). The loss or theft of digital data are consequences of this type of problems (Rangel et al. 2023, Rangel et al. 2024; Rangel et al. 2025; Rangel and Rangel 2024a). Most of these cases refer to fraudulent telephone calls or phishing, social networking platforms, bank systems, large markets or retail supply chains, fraudulent financial sector situations, and several cases of e-commerce in organizations (Álvarez 2019; Barandela et al. 2003; Barranco and Galindo 2022;

Cover and Hart 1967; Gómez et al. 2012; Javidi et al. 1997; Rangel 2022; Reddaiah 2019; Sebas 2023). In education sector, particularly E-learning platforms, there are also these types of problems. Since educational systems frequently fall behind in terms of technology and qualified employees, they are soft targets for cybercriminals (Ennu et al. 2018; Mamman et al. 2024). The E-learning platforms have a particular issue with IP theft (Mamman et al. 2024).

Despite that there exists several alternatives for handling the theft of digital data problem. We have focused over encryption methods. The encryption methods, they can use plaintext obtaining a sequence of encryption result named as ciphertext. An encryption method, it can compute statistics or mathematical equations for transforming a plaintext to ciphertext that it can only be interpreted by authorized people or systems. It is considered symmetric if only one secret key, it is employed. However, it is considered asymmetric when two keys, private and public, they are always employed. The additional secret key can be employed too. Data encryption means the hidind of information that it is carried out translating plaintext input, it is turning into a kind of encoding or language. It can use an encryption/decryption alphabet, which it will only can understand the specialized software or authorized people (Rangel et al. 2024; Rangel et al. 2023). Therefore, the inverse process, it is known as data decryption that it consists in translating a ciphertext to plaintext (Gómez et al. 2012; Paul et al. 2017; Rangel et al. 2023; Rangel et al. 2024). The plaintext is the original sequence data input. The ciphertext is the result of encryption method which it is considered dynamic when it obtains always distinct results (C_i) with the same S_i (plaintext) data input for each execution of the encryption procedure using the same parameters, otherwise it is considered static encryption schema. The cryptography (Barranco and Galindo 2022; Delman 2004; Granados 2006; Kalsi et al. 2018; Mendoza 2008; Paul et al. 2017; Reddaiah 2019; Rueda et al. 2005; Singh 2000; Susilo et al. 2021), it is considered a science that it studies the data encryption because it provides several methods, as well as, technique, strategy, algorithm, and tools which they allow the data encryption (Rangel et al. 2024). Cryptography plays an important role in the design and implementation of almost every kind of electronic system. Various cryptographic tools are required to implement security in Internet-based transactions (Drzani 2014; Mamman et al. 2024). In this regarding, we can observe in literature, several encryption proposals that they follow distinct ways of implementation (Rangel et al. 2024). Some particular cases, they are the symmetric and asymmetric encryption algorithms. There exists some algorithms of symmetric key cryptography such as DES (Data Encryption Standard), 3DES (Triple Data Encryption Standard, it is well-known as TripleDES), AES (Advanced Encryption Standard), to mention few. In this research, the AES-256 (Daemen and Rijmen 2002; NIST 2001), it has been employed. Hence, the AES (Baklaga 2024; Barranco and Galindo 2022; Bavdekar et al. 2023; Progress Software Corporation Telerik 2022; Rahman and Hossain 2021; Rodríguez 2020; Singh and Kumar 2022; Van-Tilborg 2005), it is a symmetric block cipher that operates with different block sizes and supports key lengths of 128, 192, and 256 bits (Baklaga 2024; Bavdekar et al. 2023). In contrast, the DES (Baklaga 2024; Barranco and Galindo 2022; Bavdekar et al. 2023; Dhany et al. 2018; Kumar and Sharma 2021; Mendoza 2008; Progress Software Corporation Telerik 2022; Rodríguez 2020), it processes 64 bits of plaintext to produce 64 bits of ciphertext. This operation involves substitution and permutation through a series of rounds, with decryption performed in reverse (Baklaga 2024; Bavdekar et al. 2023). However, they employment of 64 bits, it is considered insufficient for secure environments, making it relatively easy to break. As a result, the 3DES (Baklaga 2024; Bavdekar et al. 2023; Mendoza 2008; Van and Jajodia 2011; Van-Tilborg 2005), it was developed as an enhancement to DES (Baklaga 2024; Bavdekar et al. 2023). Regarding the asymmetric algorithms that they employ private and public keys such as RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), to mention few. In this work, the RSA-2048 (Rivest et al. 1978), and the ECIES-SEC (Elliptic Curve Integrated Encryption Scheme of Standards for Efficient Cryptography) P-256-R1 version, they have been only employed because they work with asymmetric key cryptography systems. In this context, they are mathematically complex. One well-known algorithm is RSA (Baklaga 2024; Barranco and Galindo 2022; Bavdekar et al. 2023; Dang and Le 2022; Luciano and Prichett 1987; Mendoza 2008; Progress Software Corporation Telerik 2022; Rahman and Hossain 2021; Rodríguez 2020), a public key cryptosystem where one key is shared publicly while the other key remains private and secret. The ECIES SECP-256-R1 (Baker and Schiller 2015; Baklaga 2024; Bavdekar et al. 2023; NIST 2013), it is a derivative of ECC (Hankerson et al. 2004; Montgomery 1987), it has emerged as a promising alternative, particularly in cryptographic

applications, due to its efficiency in addressing the logarithm problem in finite fields. While ECC and RSA are similar, they differ in that ECC uses a different cryptographic algorithm and it has a faster solution capacity. Since discrete logarithms and integer factorization are mathematical challenges, traditional encryption protocols like RSA and ECC are projected to be easily cracked by quantum algorithms like Shor's algorithm (Bavdekar et al. 2023). On the other hand, there are some standard encryption alternatives that they have been threatened by quantum computing (Baklaga 2024; Bavdekar et al. 2023; Iavich et al. 2024) such as RSA and ECC algorithms. In contrast, others authors (Fuegner 2024; Sengupta and Ghosh 2023; Thakur and Kumar 2011) warn that the AES and RSA algorithms have been also threatened by quantum computing (Baklaga 2024; Iavich et al. 2024) advent. As pointed out by some authors, dynamic encryption methodologies, they offer a good alternative to increasing noisy and redundant information in ciphertext outputs (Rangel et al. 2023; Rangel and Rangel 2024a). Moreover, the employment of noisy injection strategies, they are recommended as good encryption alternatives. The noisy injection means that a plaintext or ciphertext contains some additional ASCII (or UTF-8) characters which they are not part of the original input message. Therefore, a hypothesis of this research paper considers that these strategies may confuse to the cybercriminals (Rangel et al. 2025). Hence, the noisy injection with pseudo-hexadecimal encoding (Rangel et al. 2023; Rangel et al. 2024), it has been here recommended for being applied over plaintext because it is good indicator of the dynamical encryption performance in these domains, although that they have not been studied by quantum computing to determine vulnerabilities. It has been studied in situations that they were caused because at least one of the inadequate encryption method, it has been employed (Rangel et al. 2025). On the other hand, there exists cryptography based on artificial intelligence (Dang and Le 2022; Delman 2004; Iyengar et al. 2021; Kalsi et al. 2018; Liu and Wang 2022; Mendoza 2008; Rangel and Rangel 2024a; Reddaiah 2019; Sebas 2023; Singh et al. 2021). The artificial intelligence (AI), it has as purpose doing that the machine thinks (Rangel 2002; Rangel 2022; Rangel et al. 2024). In this regarding, there are several proposals such as heuristic methods (Rangel 2002; Rangel et al. 2024), that they consist in a previously defined set of rules for solving a problem. For example, the graph (Hartmann 2020; Russell and Norvig 2020; Sánchez et al. 1997), decision tree (Mitchell 2020; Ross-Quinlan 1993; Russell and Norvig 2020), to mention few. Other strategies based on AI, they are the random methodologies (Kuncheva and Jain 1999; Murphy 2022; Reddaiah 2016; Skalak 1994). It consists in the selection of random numbers without replacement or with replacement (i.e., it does not repeat or it repeats values, respectively). The statistical methods (Bishop 2022; Kanal 1963), they are also used in AI in several applications such as pattern recognition (Bishop 2022; Johns 1961; Rangel 2002; Sebestyen 1962), machine learning (Chen and Wang 2022; Goodfellow et al. 2021; Murphy 2022; Nils-Nilson 1965), data mining (Han et al. 2022; Hand et al. 2001; Tan et al. 2022; Zhang and Yang 2022), deep learning (Goodfellow et al. 2016; Krizhevsky et al. 2012; LeCun et al. 2015; Zhang and Yang 2022), genetic algorithms (Clark 1994; Delman 2004; Griindlingh and Van-Vuuren 2002; Kalsi et al. 2018; Kuncheva and Jain 1999; Matthews 1993; Rangel et al. 2023; Rangel et al. 2024; Reddaiah 2019; Sebas 2023; Skalak 1994), to mention few. In these schemes, it can be employed a classifier o decision rule. Some of the most popular strategies, they are considered the artificial neural networks (Bruzzone and Serpico 1997; Goodfellow et al. 2021; Murphy 2022), and the nearest neighbor rule (Cover and Hart 1967; Rangel 2002; Rangel 2022), it is named here as 1-NN (Rangel and Rangel 2024b), to mention few. Regarding the pseudo-hexadecimal encryption proposal, named here as noisy random pseudo-hexadecimal alternative. It is a method based on random methodologies. It is a derivative of random Caesar II (Rangel et al. 2023; Rangel and Rangel 2024a), with artificial intelligence because the first version, it was named as "noised" random pseudo-hexadecimal GAs (Rangel et al. 2023; Rangel et al. 2024), it was based on genetic algorithms (GAs) which it has employed the pseudo-hexadecimal encoding. This alternative has as purpose the noisy injection of the ciphertext. Due to that this proposal, it was too slow. Moreover, it has reported too many errors in encryption procedure (Rangel et al. 2024), it was enhanced as noisy random pseudo-hexadecimal (without GAs). This novel encryption procedure, it does not employ full GAs process. The GAs phases, they consist in four stages such as random selection, crossover, mutation, and evaluation using a fitness function (Rangel et al. 2024). In the noisy random pseudo-hexadecimal procedure only random selection phase, it is employed, it combined with a heuristic method for creating the encryption/decryption alphabets which it is explained later. First, the decryption alphabet in hexadecimal encoding, it is defined using random

selection without replacement. Second, the encryption alphabet (i.e., xABCD), it is created in pseudo-hexadecimal format using random selection without replacement too. The pseudo-hexadecimal value, it is computed employing the concatenation function as follows: $xABCD_i$ = ('0' + char ($xABCD_i$)). Where, the char function returns ASCII value. This schema, it allows noisy injection because it can confuse to the cybercriminals. For example, the 65 value, it is the ordinal number of ASCII «A» character, and the 41 value, it is the hexadecimal number. However, the «0A» sequence, it is the pseudo-hexadecimal encoding. Hence, the «0A» value when it is transformed as hexadecimal number, it can be the ordinal 10 value of ASCII table. Therefore, it can refer to the escape sequence of new line. In these terms, the ASCII «A», it is noisy character when the «0A» pseudo-hexadecimal value, it is applied (Rangel et al. 2024). Initially, the experiments were focused on replacing of the static encryption schema for recommending the dynamic encryption methods such as novel proposals based on noisy random pseudo-hexadecimal alternatives. Of course, this situation should not put in risk the digital data security in the organizations (Rangel et al. 2025). In the present paper, several aspects related with the subject, they were studied, as well as, the average of experimental results, they are here shown. We focus on cybersecurity strategies based on four variants of noisy random pseudo-hexadecimal methodology (Rangel et al. 2023; Rangel et al. 2024), it includes the standard version. That's why that only three novel implementation of these schemes, they are here proposals. It was carried out with the purpose of suggesting the noisy injection over of the plaintext as safety measure for safeguarding the data digital of the organizations.

To facilitate the comparison of results with other authors (Rangel et al. 2023; Rangel et al. 2024; Rangel et al. 2025), only two classes of situations, they have been studied. First, the comparison of the standard encryption algorithms (i.e., DES, 3DES, AES-256, RSA-2048, and ECIES SECP-256-R1), they were applied over plaintext as safety encryption alternative. Second, four alternatives of the noisy random pseudo-hexadecimal methodology, they have been applied separately over the plaintext with noisy injection purpose because these pseudo-hexadecimal strategies, they have been rarely studied. These alternatives include standard version. Therefore, only three novel implementations, they are here shown. This schema, it has been named here as noisy random pseudo-hexadecimal strategies. These strategies are considered dynamic alternatives for increasing the digital data security (Rangel et al. 2025). The evaluation of the standard encryption algorithms, as well as, the standard noisy random pseudo-hexadecimal strategy (Rangel et al. 2023; Rangel et al. 2024), they were applied separately over plaintext in previous research with five-fold cross-validation (Rangel and Rangel 2024a; Rangel and Rangel 2024b; Rangel et al. 2024). In this work, the four noisy random pseudo-hexadecimal alternatives here presented, they have been employed in the same way clarifying that three novel modifications, they were implemented (i.e., by substitution, by shifting, and version 2). Hence, the evaluation of these dynamical encryption methods with five-fold cross-validation, they were also evaluated. In this work, the performance of those methods, it was expressed in terms of the global average or accuracy (Barandela et al. 2003). These schemes are also considered dynamic alternatives for increasing the digital data security and performance of the standard encryption algorithms. Due to that cybercriminals, they do not know where is the position of the noisy injection character or pseudo-hexadecimal value. Moreover, these proposals, they have not been yet studied for brute-force attacks (Bošnjak et al. 2018) or quantum computing (Baklaga 2024; Bavdekar et al. 2023; Fuegner 2024; Iavich et al. 2024; Sengupta and Ghosh 2023; Thakur and Kumar 2011).

Besides, several of the noisy injection ciphertext exemplars that they have been obtained by those novel alternatives based on pseudo-hexadecimal schemes, they are here shown including ciphertext of standard encryption algorithms for comparing results. Finally, we propose the noisy random pseudo-hexadecimal injection ciphertext schema because it is good indicator of the dynamical encryption performance. Of course, this situation should not put in risk the digital data security in the organizations. Novel definitions of noisy random pseudo-hexadecimal alternatives based on artificial intelligence (AI) and comparison results with several standard encryption algorithms, it is also introduced. In conclusion, the novel proposals that they have been here recommended, based on the examined ciphertext with noisy injection using pseudo-hexadecimal format, the results have disclosed that they can be of difficult decryption for cybercriminals and even the quantum computing.

2. RELATED THEORETICAL AND EXPERIMENTAL METHODS

The replacement of cybersecurity strategy, it does not guarantee the data security of the organisations (Rangel and Rangel 2024a; Rangel et al. 2025), when it consists on the employment of inadequate encryption algorithms (Barranco and Galindo 2022; Gómez et al. 2012; Mendoza 2008; Rangel and Rangel 2024a; Rodríguez 2020; Progress Software Corporation Telerik 2022). In these cases is recommended using a dynamical encryption methods which they are able to obtain distinct ciphertext as results with the same plaintext input (Rangel and Rangel 2024a; Rangel et al. 2025). First adopted strategy consists in the employment of standard encryption algorithms as data encryption measure for comparing results with our novel proposals. A second adopted strategy, it consists in the noisy injection application based on four noisy random pseudo-hexadecimal alternatives because these schemes, they are a good indicator for improving the security of ciphertext outputs (Rangel and Rangel 2024a). These schemes have been evaluated separately. Despite that the produced ciphertext exemplars, they are at least six-times bigger than plaintext or depending of the filling in final package with ASCII characters, it can be until 70% bigger than plaintext. This paper describes four alternatives based on noisy random pseudo-hexadecimal methodology for handling to the theft of digital data problem (Rangel and Rangel 2024a; Rangel et al. 2025). In this research, only three of them are novel implementations of the previously named in (Rangel et al. 2023; Rangel et al. 2024) as "noised" random pseudo-hexadecimal schema which it is here named now as noisy random pseudo-hexadecimal standard alternative. Therefore, the novel modifications, they have been also named as noisy random pseudo-hexadecimal methodologies. Therefore, they were applied over plaintext as a noisy injection strategy.

First, adopted strategy consists in the employment of traditional version that it is named here as the *standard noisy random pseudo-hexadecimal* alternative, it can be computed as follows: *finalPackage1 = noisyRandomTraditionalPseudoHexadecimal* $_{(i+j)}$ = (($ABCD_i + R_j$) & ($xABCD_i + R_j$) & ($C_t + R_q$)). Where: The C_t vector is *ciphertext* of the S_t *plaintext* with size of *t*. It is obtained as follows: *If some of* (S_t == $ABCD_i$) *then* $C_t = xABCD_i$; *otherwise* $C_t = S_t$; *end-if*. Therefore, the *ABCD* and xABCD vectors, they are the *decryption* and *encryption alphabets* respectively with size of *i=120*, due to *mod 120* that it has been employed with a range of the *ASCII* table between *30* and *150* values. The R_j and R_q parameters, they contain the *ASCII filling* characters with *hexadecimal encoding* that they have been selected randomly *without replacement*. Therefore, *if (t > i) then j = (t-i); q=0; otherwise q = (i-t); j = 0; end-if*. The + operator is *concatenation function* while the *&* operator means the *interspersed concatenation* as follows: ($ABCD_0 + xABCD_0 + C_0$) + ($ABCD_1 + xABCD_1 + C_1$) + .. + ($ABCD_{i+j} + xABCD_{i+j} + C_{t+q}$). Finally, the $ABCD_i$ and $xABCD_i$ alphabets, they are selected randomly *without replacement*. When the *artificial intelligence* has been employed with a *heuristic method* very similarity to the *selection process* of the *genetic algorithms*. First, in *heuristic method* the *_ABC* vector, it has been defined as *validation method* of the *ASCII* characters that they can be introduced as *plaintext* and they can be also assigned in the *alphabeths*. In this research, the *_ABC = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', '°', 'y', 'z', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '¶']*, it has been employed. The *'¶'* character, it can be replaced or being included other *ASCII* additional values that they might be *noisy injection* or only other *validated character* of the *plaintext* or *alphabets*. Hence, the *ABCD* and *xABCD* vectors, they are selected randomly *without replacement* using *_ABC* information with *pseudo-hexadecimal* encoding. Each time that $_ABC_i$ character, it is added in *ABCD* or *xABCD* , this *ASCII* value, it must be removed from *_ABC*. Therefore, the *selection procedure* for the *ABCD* and *xABCD alphabets*, it must be carried out separately. Second, when the *_ABC* vector, it has been emptied the *ABCD* and *xABCD alphabets*, they must be *filled randomly* with *hexadecimal* values with a range of *mod 120* (Rangel and Rangel 2024a; Rangel et al. 2023; Rangel et al. 2024). This *_ABC* vector, it must be randomly dispersed in both vectors: *ABCD* and *xABCD without redundancy*. The *pseudo-hexadecimal* encoding, it was here computed as follows: $_ABC_i$ = ('0' + *char* ($_ABC_i$)). The + operator is *concatenation function* while *char* statement, it means that the *ASCII* value has been obtained with *mod 120*.

Second adopted strategy, it consists in the modification of standard noisy random pseudo-hexadecimal schema, which it is named here as the noisy random pseudo-hexadecimal by shifting implementation. It has been recommended as novel proposal. This alternative can be computed as follows:

finalPackage2 = noisyRandomShiftingPseudoHexadecimal $_{(i+j)}$ = (*Hex* ($ABCD_i + R_j$) *& PsH* ($xABCD_i + R_j$) *& PsH* ($C_t + R_q$)). Where: The $ABCD_i$ vector with *mod 120*, it contains the *decryption alphabet* with *hexadecimal encoding*. This vector has been selected randomly *without replacement* using a process very similarity to the *selection procedure* of the *genetic algorithms* (Rangel et al. 2024). However, the $xABCD_i$ vector with *mod 120*, it must be obtained *without replacement* too, due to that it contains the encryption alphabet with *pseudo-hexadecimal* encoding. This vector can be computed as follows: $xABCD_i = ABCDouble_{(i+w)}$. In this case, the *ABCDouble* vector, it must be assigned as follows: *ABCDouble* = [*ABCD* , *ABCD*]. This operation in *Python* language, it can be computed as follows: *ABCDouble* = (*ABCD* + *ABCD*). The *w* parameter, it must be assigned as follows: $w = K_i$. Therefore, the ordinal K_i vector, it must contain only distinct random shifting values for each $ABCD_i$ position of the *alphabet*, they were selected *with replacement* and they are limited with a range between *0* and *120* values. Consequently, the C_t vector with *pseudo-hexadecimal* encoding, it is *ciphertext* of the S_t *plaintext*. Hence, the $C_t = xABCD_i$, it must be computed only *if* ($S_t == ABCD_i$), *otherwise* $C_t = S_t$, it must be assigned. The first R_j vector of *finalPackage2*, it contains the *ASCII filling* characters with *hexadecimal encoding* while the second R_j and R_q vectors, they have a *pseudo-hexadecimal encoding*. These cases have been selected randomly *without replacement*, while they K_i vector, it is selected *with replacement*. In the same way that the *standard noisy random pseudo-hexadecimal* version, the operations are carried out. Hence, *if (t > i) then j = (t-i); q=0 ; otherwise q = (i-t); j = 0; end-if*. Finally, the *Hex* and *PsH* procedures, they correspond to the *converter functions*. Thereby, they return *hexadecimal* and *pseudo-hexadecimal* values, respectively. These *pseudo-hexadecimal* values, they can be computed as follows: $PsH(arg_i)$ = ('0' + *char* (arg_i)). The arg_i *parameter*, it corresponds to the *ASCII* character of *alphabet* or *filling* vector.

A third adopted strategy, it is named here as the *noisy random pseudo-hexadecimal by substitution* alternative. It is also a novel proposal based on *noisy random pseudo-hexadecimal* standard version. It has been implemented as follows: *finalPackage3 = noisyRandomSubstitutionPseudoHexadecimal* $_{(i+j)}$ = (*Hex* ($ABCD_i + R_j$) *& PsH* ($xABCD_i + R_j$) *& PsH* ($C_t + R_q$)). Where: The $ABCD_i$ and $xABCD_i$, both with *mod 120*, they have been selected randomly *without replacement* very similarity to the *selection procedure* of the *genetic algorithms*. Therefore, *if comparison of* (*char*($ABCD_i$) == *char*($xABCD_i$)) >= *10*, *then* these *selection procedure*, it must restart because this implementation does not allow *redundancy* in more than ten characters. The $ABCD_i$ vector is *decryption alphabet* with *hexadecimal encoding* while $xABCD_i$ is the *encryption alphabet* with *pseudo-hexadecimal encoding*. The R_q and second R_j vectors, they contain the *ASCII filling* characters with *pseudo-hexadecimal encoding* while that first R_j vector, it contains *hexadecimal encoding*. The C_t vector with *pseudo-hexadecimal encoding*, it is *ciphertext* of the S_t *plaintext*. The C_t vector can be computed as follows: *if some of* ($S_t == ABCD_i$) *then* $C_t = xABCD_i$; *otherwise* $C_t = S_t$; *end-if*. The *pseudo-hexadecimal* values, they can be computed as follows: $PsH(arg_i)$ = ('0' + *char* (arg_i)). The arg_i *parameter*, it corresponds to the *ASCII* character of *alphabet* or *filling* vector.

Fourth adopted strategy, it is also a novel proposal that it has been named here as *noisy random pseudo-hexadecimal II* alternative *(version 2).* This implementation can be computed as follows: *finalPackage4 = noisyRandomPseudoHexadecimalVer2* $_{(i+j)}$ = (*Hex* ($ABCD_i + R_j$) *& PsH* ($xABCD_i + R_j$) *& PsH* ($C_t + R_q$)). We can observe that it follows the same way as the *noisy random pseudo-hexadecimal by substitution* implementation. The difference between both alternatives, it consists that in the novel *version 2*, the $ABCD_i$ and $xABCD_i$ vectors with *mod 120*, they have been selected randomly *without replacement* very similarity to the *selection procedure* of the *genetic algorithms*, but it does not verify if it occurs *redundancy*.

On the other hand, we can observe in several of the *encryption final packages* that two *alphabets* and the *partial ciphertext*, they have been only included in these packages. However, the K_i vector of the proposal based on *noisy random pseudo-hexadecimal by shifting* implementation, it has not been saved in *final package* because its encryption C_i vector, it was obtained with *substitution procedure*. Besides, the information has been camouflaged as *ASCII* character being converted to *hexadecimal* or *pseudo-hexadecimal encoding*. This situation can help confusing to the *cybercriminals* and it avoids putting in risk the data security in organizations (Rangel et al. 2024; Rangel and Rangel 2024a). In

this context, there exists a recent research (Rangel et al. 2025; Rangel and Rangel 2024a), which it has revealed that *noisy injection* based on *reduced and random* schemes, they are good alternatives for safeguarding digital data in organizations. In this same way, we have implemented three novel modifications of the *standard "noised" random pseudo-hexadecimal* schema (Rangel et al. 2024), that they are named here as *noisy random pseudo-hexadecimal* alternatives. These methodologies have been evaluated separately. They were applied over *plaintext* as *noisy injection* strategy. Hence, the employment of these *noisy random methodologies*, for the encryption of information, they have shown to obtain *dynamical ciphertext* results, which it is good for the data security. Nevertheless, this situation can produce too many errors when some sequence of characters, it is out of range to the *ASCII* table (Rangel et al. 2023; Rangel et al. 2024), due to the random K_i vector, it is not delimited (Rangel and Rangel 2024a).
Regarding the evaluation of results, in this research, for internally biasing the discrimination procedure, it was proposed the cross-validation method (Barandela et al. 2003; Rangel et al. 2025). This information can help suggesting the employment of noisy injection as safety measure in the organisations. Of course, additional disk storage space, it should be sacrificed to implement the proposals here recommended.

3. RESULTS AND DISCUSSION

All the experiments were carried out over five random samples of *ciphertext*. The *execution of times* with *estimated error*, they were computed for each *encryption method*, separately. If the result of the *ciphertext* when it has been applied *decryption process*, it is not equal than *plaintext* input, it was considered an error. These samples have been evaluated with a modification of *cross-validation* method (Rangel and Rangel 2024a), it was computed for facilitating the comparison results with other published research (Rangel et al. 2023; Rangel et al. 2024; Rangel et al. 2025; Rangel & Rangel 2024a; Rangel and Rangel 2024b). Most of the experiments with *standard encryption algorithms*, they were developed with *Python 3* (Python.org 2024). Other tests were executed over *artificial training samples* which they include *ciphertext*, the *encryption time*, and *computed error*, they were applied using personal computer with *4 GB* of memory *(RAM)*, *CPU* frequency of *1.2 GHz*, free disk space of *2 GB*, and *Windows 10* operative system for comparing with other authors (Rangel et al. 2025; Rangel and Rangel 2024a). In addition, several experiments, they were carried out using *Pydroid3* with mobile computing application (Pydroid3 2024), it was using *Android-9* operative system of the same *hardware features* of the *personal computer* above-mentioned for comparing results. In the same way, the four *pseudo-hexadecimal noisy injection* alternatives, they were implemented with *Python* language.
First, in *Table 1*, the *encryption/decryption* procedure with original input of *plaintext* over five random samples, they were applied using *standard encryption algorithms* for being compared with the *standard noisy random pseudo-hexadecimal* alternative, separately. Regarding the application of the *standard encryption methods*, they were applied using the *parameter* such as it is described as follows: The employed *parameters* for the *DES* algorithm, they were considered a *key of 56 bits* with the "*00000001*" value, as well as, the *ECB* (*Electronic Codebook Mode*) format, and *hexadecimal encoding* (*UTF-8*) for the *ciphertext*. The developing of the *DES* algorithm, it was carried out with *Crypto.Cipher* (PyCryptodome 2025), package of *Python* language. In case of the *TripleDES* (*3DES*) algorithm, the *ECB*, and *secret key of 24 bits* with "*000000000000000000000001*" values, they have been employed. Being obtained the *ciphertext* results with *hexadecimal encoding*. The implementation of the *3DES* algorithm, it was developed with *Python* language using the *Crypto.Cipher* (PyCryptodome 2025), package too. The experiments with *AES-256*, the *Cipher Block Chaining* (*CBC*) mode with *IV* = "*0000000000000001*" vector, and *Public Key Cryptography Standard #7* (*PKCS7*) with *256 bits* of *padding*, they have been employed using *hexadecimal encoding* for the obtained *ciphertext* results, this with a key size of *256 bits*, as well as, the *k=32* for being executed in the "*Crypto.Cipher(algorithms.AES(k))*" statement using the *Crypto.Cipher* package of *Python* language. The *encryption/decryption procedure*, it was applied with *secret key* value of "*00000000000000000000000000000001*". Concerning to *RSA-2048* algorithm experiments, the *Privacy-Enhanced Mail* (*PEM*) format with a *size of 2048* for definition of *public* and *private keys* with *secret key* value of "*00000000000000000000000000000001*", they have been employed. In the

encryption/decryption procedure, it was carried out with a *padding=16* for the *Public Key Cryptography Standard with Optimal Asymmetric Encryption Padding* (*PKCS1_OAEP*), when the *secret key* value of "*00000000000000000000000000000001*", it has been employed with *hexadecimal encoding* of the *ciphertext*, it has been obtained. During the *RSA-2048* experiments, the *public key* was employed as follows: "*-----BEGIN PUBLIC KEY----- Proc-Type: 4,ENCRYPTED DEK-Info: DES-EDE3-CBC,E03C091242BA1BF3*
FeCI/5KgBXgbnQ8T8bV9DIlAUQ7JDah/nKpNJinT5gCpNl9RD/RGP5JA9hTdqaq/GwBrCioaP24tHBDJ2icLDx5qJjrDKh4
a3B36JeC49PZK6gNTyGy6WdMXCk4h599YYnhVFyWjyPoQTHYgQk2bQkK2/+NAQWkLiDW29V8pr+xhqGn0rN7oPOkP
BSY/XZ5CWb/wstqnTeDB7VmiGPClDBnsYv26bwnpCI5G/iz/okYdgvGlpsOtyTd3P3OYH/4C9/tDraA9UEeACYQ5DsBAmsA
ncgeY6js+rGz7xS9cyYfYVuqmFrNKABSxtT81TFhzfXjQ2fcROvQOeMNBAKe/UbQQVXsE1hJ/zynnc9n0KtVGk1AtLGzglzK
YCewiB34JQ4D7W/40tK0= -----END PUBLIC KEY-----". Regarding *private key*, it was employed as follows: "*-----BEGIN RSA PRIVATE KEY----- Proc-Type: 4,ENCRYPTED DEK-Info: DES-EDE3-CBC,FEA0B6E4A8048D2B*
JjSA7yUk5aP6oQqi4AOF36ACF0q1oxcpqK5tttPhhl9i5g9X/wVirndpgs34SP+k
ElEIqOOBfN4BZtuQnSqkzwOPOzb2Z9VPlCzmCPVKuka9V+lPDKVdoOtpdzAkyK+8PRbGo3V4ksFUdzhBfaJfdGRBjQ3OEtP4H/i4HygZa
Semxnfmxwgq9ELhHJ+yX8uKtBjn/O50pfKnMQGSZ+pWNa/dR2nsEkbGueUKYyJeWy+aNAO7O146MUVXFHTW6sMyJNTJs7AHyJ47
Ms7Z09Ps7Q7x0S1PHY3XiMb4UyX050V3+5c2P9y+2nA9YRhliUi93LIJ6oGhhJhu/VtCkYPSGTau/bu+749gDmN5hUP+U++eHqZ3cxYm9
pM1J5iyFQBBNSq7kuk2O4Uw8iABGPjmoiyckIY91udylCqS6FZfqq241fGruR00Q73esSM3QH+wh/IEnFvCOjpUMkYRSyUHA2XvuFO9zas
sf3kMv/vqI03qf/6vu4mfwlnfgZigYEIfTUub14wlS82OKieBVtoON8jHXySzrnwmE5WFANjWKlgRThQRE7K36JhMcIRBk17Xt0e6xRyj/Il5CxB
/FDBGBebp8iubbGPEAKEiUMuu4ZwBj5VMpVTShcsoMwFS4n4v1qALS+BpZM+FBH5NqoCkW/dE/GHlm16uDd8bSVAODcEOsiB85Na
M2UOsU06n2pWv8wdS9yG8FKqpEJJVDsYka9xtOZ7jA6zRYQIW+OMcyPQW/MkTIj1Sy9+7YeSQSYL6j2tGniTi2d8mn4qOEYY7rimsWEw
qOcx2SnvELZAo8t+rFr/AginnOyHAKhSdHsKC7l42/Gtt4KEyI1viUEOXHD2HtxJ8/mQYJ3om7xBdDAj1F4bP/Li2ZMuBXHV3gI6z2TZCqjit
tHC0AmBGMhFuc2PggLWC+wwkXteBB+jpe1ljmdfqOAxekH9zhPSDD5geyNOLYbUQag/V1Idppj+wI1noBJeni53Z35V4xrMoMAwMR+R
GSu496eYr61sAhpgCnGtR4XKmf7L3XiFCUCnJ0JGGQtKd3QPdmiiW2xpTI2xyvdcFVUL0MEAXyFOMcugfwF+nhwtMxAIynvgKBO5dRK
5KknsKiDUsPpBwoaIdZ9s+BZ5fdYqIjFrk8MjUFsHn1CiMop+BT6fCHjACHjEzZbUhDVrNOSBucxwqnK7b+JAApK9quGjFCyclTViflHPA
GlONMjhp9f/kWVvh2e09SrCte4jFVKQS7RXwHsBFD/gP0EU7DrfRtqGbqrvmnrGofC5blgBdY8IV8QFtTG0e+tq/csMURnHsbb+DKF149/J
gz1wE+2myUJXcomi42n8iAynX+I1v/q3XHyo7CmzLjDr8us3pIef7YB3WoyAcbrjG+cQZlKhQfUXnrQL6M1EGT1n3Dlq1YFJbgiLj6EpC9LY
xC4SwKJNoat31yaMWADorwZhjMDknr64KhaRUSDEF5foVfnyCGZzZ4vT3AjOy0uGBugxrf3DBra1nGA1tK4RbDHlo9njPOAaL5oQIfH5i
p39XEL1rG9gPocxpPhgMRBgmTWMkNYDoRsvG30rly6JjlMCXnUWJg==-----END RSA PRIVATE KEY-----". Both keys, they were created using the *Crypto.Cipher* and *Crypto.PublicKey* (PyCryptodome 2025; PyPI 2024) of *pycryptodome* (PyPI 2024) library. Finally, the experiments with *ECIES-SECP256R1*, they were carried out using the *Privacy-Enhanced Mail* (*PEM*) format with *size of 256 bits* in the definition of *Public/Private Key*, when the *Public Key Cryptography Standard #8* (*PKCS#8*), and *secret key* value of "*00000000000000000000000000000001*", they have been employed. The *ciphertext* was obtained with *hexadecimal encoding*. During the experiments, they have been employed using a *password* or *secret key* value of "*00000000000000000000000000000001*". Moreover, it has been employed the *ethernet key* as *public key* with value of "*0xd28eca4fe0f331080dec594a020429c041d78ce8d3eedc49cc523fd9fb42393761d34af640166a6e794b50733e5*
1a2ac68cb7720c2b88cb096dc120877261567". Being derivative of the *public key* as follows: "*-----BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE9UVFCN2r+*
Mszac1YOWxY5+AjxI7+s2vxuyo2oykAr0uyvawcimw2RQsvrrtrjZY2rpRCYiZty2WVXJyEeXOVGw== -----END PUBLIC KEY-----". Regarding the *ethernet private key* that it was employed with the value of "*0xd7a530e15ed1af8515973d43a0fe9c67cdb75fb8345704593d9b0dae57647726*". It was derivative of *private key* as follows: "*-----BEGIN ENCRYPTED PRIVATE KEY-----*
MIHsMFcGCSqGSIb3DQEFDTBKMCkGCSqGSIb3DQEFDDAcBAioxkswfb4AygICCAAwDAYIKoZIhvcNAgkFADAdBglgh
kgBZQMEASoEEHqWiuvsdqRIsX3PJ6AXKWEEgZCDvif21Y+esz0Q8/LZeWAsMcH6zuolejJhzcIjjBOjl4POufmYlSq87O/Hg
R61IEnGdHUU5zoaerGLCI26Z6wDhbZv6tyJnLRT/lrzdia3KiaKF4y+ne2OnRaMbX6MmASU3RX312wyYWREcHHiUdhJJY
H99cDuBoc6Rcgjmz3aA555JsqBzh0N+WRumxhlbm8= -----END ENCRYPTED PRIVATE KEY----". The *public* and *private keys*, they were created using *standard values* that they have been obtained by *eciespy package* (ECIESPy 2025) of *Python* library.

Second, three modifications of the *standard noisy random pseudo-hexadecimal* schema, they were implemented. Moreover, these novel alternatives, they were applied over the same five samples with only five repetitions (see *Table 1*), separately. This operation allows the comparison result with standard encryption algorithms.

Finally, the *compatibility* of *noisy random pseudo-hexadecimal* alternatives, they were also evaluated, separately (see *Table 2*).

Tab.1. The calculation of encryption/decryption times (in milliseconds) and estimated error with original plaintext, they were computed. In columns A and B, the average and standard deviation values, they are here shown, respectively. The ciphertext results for each encryption method, they are shown too. Two experimental tests with the same plaintext: "▒Welcome▒□", in some cases, different results have been obtained.

	Ciphertext		Encryption time		Decryption time		% Error	
Encryption method	**Test 1**	**Test 2**	A	B	A	B	A	
DES	*d535c337be1d94db5b940b75bce124dcda93ffa35791bf07*	*d535c337be1d94db5b940b75bce124dcda93ffa35791bf07*	1.1440	0.5087	0.1937	0.0187	1	0
3DES	*d535c337be1d94db5b940b75bce124dc620c51e2380a3d5c*	*d535c337be1d94db5b940b75bce124dc620c51e2380a3d5c*	7.6814	3.9388	0.5005	0.0184	0	0
AES-256	*3030303030303030303030303030303030319421780fc40c59f14796a598115d8e5139f6f9666f67cb30fcda01f22739eeda*	*3030303030303030303030303030303030319421780fc40c59f14796a598115d8e5139f6f9666f67cb30fcda01f22739eeda*	7.3117	3.7093	0.6166	0.0169	0	0
RSA-2048	*2ff1af10168db5afd122840eeba316005d4663edfabb1d68763f10d8739e5556576775042033d040e2508082202bed5066792ae01f9cd046d971f297a4402c2d240ee17847acf1db9441ee700d54e351e09e76c7f5b7b7a120f9ea1cb36c9fe17fd586af047bb96e8d0f6a7249003cee2df97d62678f844eac1cd7c688b1e4609ba55b5099f0236e666a63cbbc65dfb1d3f9805a9e79de8370fd2148db467fcf300d3ec3fc491da2a488a26e3a1f1d41bb8c7b3005e1805139e295f6a67448ffac8254067eb9d19905fe36601b0b6f81df02c35a5e00d063d5b438465896d967f7433dfcb88b0f4249a47459a5c91a8b56c86f0ce0de44779be90369a79f7cb1*	*838e1934814e9d4656f44ca0f82a659c0630862d26f79e78e0b37da4e1ddb265db0567abd650884b71fdba8ed3161e75209046cf4f9ca6f20c53fcdb48e8d8e4767b6d9cde89e3a95409f21f7b5badbc2537690b7ad06fb4068ef50d9284495a3e4f01cfc448ae13848d7b165e6a9ae30062c117112e80dc57602a01c554cb8e9809b979cce600edb2adcac96896ca08886193d1ed6fc271e1a59d218da9406dc0e0a6e16692c404db8ef3dc39938054cf29df4b70b1c468842ff2a017d5f095aa234a5e28b4991e46de7f0e52dd01eaaf7b3f47c3f49feafea596ac73951641e7ac55af38e01d3f118f356ef50278ee77b9769ba0f6688a6118c1fc0ed9121*	12.2807	0.8885	355.2075	0.7505	0	0
ECIES SECP-256-R1	*047e667dd985719abe0e896647b01ddb57d7eb960b75408134714e0cdc80abd47bb1d8232b6e46aa16f1b3ad527823b96d45a13fff835be6e984f9da7a585d5075da2a3cb414417142b02d069d56549641044485e559dae974c634ac6a378b56223921dd55ab96e8d7912cd9606abc50f306b*	*0408d156e9f10de4af9663961d6e0fb0b434390ad039caad7de2e45f784122788c3cd740dec5822e74f782ee2e22ab6695f3866c9f78258200a280e84625a061776d4432e20b9d220da8ae899a45cf5576e42c72bb7f6157a09119cfac671f2fbb70e7a786ca1dbd11aac38259008adfd145*	4.8233	0.8422	3.5989	0.0927	0	0

"Noised" random pseudo-hexadecimal GAs (Rangel et al. 2024)	6d0-0▓940X0«7900]530_0l 5900G20050400ë0-370@0] 90030▓7e00280'fd8d0:ff8f 0$fd5a0fe350fdcd0fe5c0ffea0.f fb609fe520Àfd310fd8e0 fd480Yff4707fe160%fe360E ff690ff260Sfe560qfe430Ufd830 Pfd1f0ff750\|fd670Jfd730ff5d0" fe1c0tff3d0>fe8b0wfd3a0fd85 0fe800Tfe410,fd2f0Nff290afd4 c0;fe910?fe500!fe390feb20Ffe 7a0fe4a0[fe630Gfd740mfd3f0f f7c0jff600offb70Dfd610iff2d0& fd5b0dfe880ff3b00fe5f0#fd810 ~fd660Lfdd00pfd2101fd230gff 340fd510Bffc00Vfd6a0+ff890M fe640*fe3e0rfe4b0sfe380Wff6 b0^fd8c04fe460yfd7208fd7b0/ ff220ff570«ff7d0Cff700Qfd680 eff2c0fd870fd5e0Hff840zfd7f0 xfe710}fe250=fe550Åfd920ufd 760(ff2b0fe4e0fd990\fe770<ff 450Aff950`fd240Ofe1e0kff4f0b ff620{ff540fe2e0¢fd4d0Zfd650] fe320)fd6f0fd4206fd300Kfe930 ffd860hfd090fe820Rff | 2700▓450C0>460u0p4c0~0E6 200^390*0/940.0Q4200p2b0Z 0▓3d0#07706fd280fe5c0ff82 0fe560Aff340Nfd600×fd8e0hf d670)fe430Kfe330wff8c0ff540 ff6c0Efe7e0Sfd6a09fe260_ff6 10efe210fe3f0Jfd2a0Tfe750\|f d480}fec50fe410[fe5b0"fe4a0' ff890fd4d0`fd3d0Bff920ofd78 0,fe630^fd6b0ff2d0afd950<f e590Iff440bff570>ff350Ûff 490kfd8b0Mfd360Yfd700nfe3 80ff310Rfd2c0Lfd9103fd470sf d4f0ffe3c0+fe7d0xfd3b0$fe6d 0Qfe790dff240Ufe810fe500m ff7f0Hfd7c0fd6408fd230%ff86 05fe720Ffd4b0Pfe830@ff7b0 4ff1f0vfd5f0ff200Gfe580{fe37 0fd6e0ff2e0rfd800Xfdba0&fe 4e0fd660-fd220gf e290fe8a0j fd3a0lfd5e0fe930qfe900!f d8f0(fd730\ff690Âff3e0fe5a0f e650pfe6f0/fe710;fe880yff76 0?fd550ife320ñfe5d0Dff1e0W fe520=fd2500fe400fd7a0Vfe8 50Off870tfd740ff2f0]fd300fd6 80cfe510zff	2467.9564	796.0514	12.2143	0.3905	0.20	(
Noisy random pseudo-hexadecimal GAs (our experiments)	770W0▓700w0⍰6a0y0½8c0t0⍰ 210%0620V0⍰290Z0,3b0_0½5a 0r0▓4d0þ0□5d0k0860-fe3f0PfeSc0ðff4f0⍰fe810{fe 450Sfd830`fe4b0)ff260⍰fe930H fd500[fd2d0pfd530*fd300Ñfd7 60fff4a0⍰ff4c0:fe560zfe320qfd 510Cfe8e0⍰ff270hfe200Üff790 ⍰fe920«ff6e0.ff820Gfd910,,fe3 a0Nff6105fd2f0|fd890‘‘fd5b0cf e1f0Àfd8d08fd900Iff880~ff7c0 ⍰fd7e0#fe590Aff380?ff480Yff8 50xfd800bfd640jff350"fe470Xff 5e0•ff430ff020vfe940lfd2c0Qff 840iff780Ufe660Šff490ñfe7d0e fe7501fd720Rfe740fd7f0;ff 630ff1e0Off4603fe8f0Öfd390u fe580&ff690Mfe6f0⍰ff520⍰fd3 d0‡fd600+fe4e0aff8b0nfe410⍰ ff4402fd2e0<fd870!fe550Jff01 0‹fd650½fd3e0/ff280gfe2b0Tfe 0a0ff400Bfd710…fd240\ff8a0† fe680^ff0e0,fe420dfd6c0‰fe2 20fe960Œfd1800ff3607ff670sf e230]ff950®fd7a0$fd570˜fe3c 0Kfd6d0,fd370mfe330>fe340of e310Lfe250'ff | 340i0▓8a0⍰0g320X0I3e0S0~f 10⍰02a0ð0k880⍰044c00I250F 0▓6c0~0□7e0(07307fe700qfe 230Zfd3a0lff390'fd670{fd460= fe840¼fe210⍰fd8b0Aff350*fd 910pfd930efe6e0$fe530Cfd41 06ff590⍰fd830Off5c0>ff300⍰ff 4b03fe5a0:fe7c0.ff6b0!ff770|f e760⍰ff330fdd20sfe450⍰fe3c0 jfd710⍰fd360Qff490+fe2b0⍰ff 690afd290wff380\fe780•fd75 0nff6f0kfd7f02ff8f0dfedb0⍰ff 950[fd1e0#fd2c0ofe580)fe8e0 Dfe5d0‘‘ff790vfe640@fd7b05f e280Kfd200Šff5f0fd4f0_ff2e0, fd860Hfe260,fe3b0cff1f0`fd80 0ñfe5e0&fe7a0Mff600Ùff2f0] fd5201fd920‰fe2d0Gfd550Ef d3d0xfd7d0Rfd680Jfd900tfd6 30ff4a0Ÿfe400/fe6d04ff650Ife940?fe720bfd500Ufe570gfe8 90rff4e0-fe240Vfd510Nf f740^fe220‡fe480;fe6a0fff420 <fe270"ff820˚ff3100fe470hff6 60†fd560uff5b0Pff3f0Lfe440⍰ ff8c0yff5409fd8508feb00mfd8 10ûfe4d0ff610}fd870Yfd	3153.6003	259.8847	12.5324	0.7191	0.32	(

Noisy random pseudo-hexadecimal I	*85830▒73584e3f47310Z2e 782f7f5e0U0r0H835c6a050631 0X0Ñ0▒0n590w0u0T0610Ufff0 L00ff6f08ff0y68fe5e0Vfe3d0pfe 0m6afd842cfd8261ff0F94fe0i8d ff2b43fd0k0Kfd897dfd5b0qfd52 60ff0s49ff0g3afd0e31ff0¶7efe0 ñ22fe4f0bfe3237fd630Nfe650Df f0v0Qfd8069fe0j0Afe3b0¶fe0l7 8ff2d02fe0H87fd0□0wfd4050fd 0D3dfd452bfd0J0Ofe010ifd090 vff530Jfd0r0Eff3363fe673efe0b 25ff5c05ff0d0nfe0Ñ0ufd0M0zfd 0E0dfe0I0sfe290Xfdb60lff0f0Rf f0B81fe4e0off8a6fff0Q3bfd0C0 afd410yff0S0□ff0p70fe496dff6b 0Cfe312aff550tfe030ñff0o0Hfd 0W4efe0t0Bfe2a79fd0P0cff870 kfd0G92ff930efe940Ifd040gff0 w4bfd060Yff910jfe082dfd8604f d0q07ff6493fe0Y75fe0R0ffe7c2f ff370mfd0c5efd0N57fe2c80fd0 V0Gfd0A89fd2203fd6d0hfe0O2 1ff0K0Mfd0T54ff710Wfe020Sfe 2446fe680Zfd0001fe3e7cfd0h0 9fd070Pff8d0Fff0a8afe7e6bfd0 z0Lff*	*0g050▒090V0a26220M0T073 23f0c0o0A30530H0s620¶8f0 Mb6010▒0q0ñ650D0Z00c0of e000Gfd6976fd0Y0¶fe550Bfe2 825fd7736fe3b0nfe8d0Cff0N5 5fd0E0Rff7c44fd2c0Jfe6f31fd0 w2eff6608fd0U0Lfe0M5efe8e0 Yfe0141fd306cfd5e2fff0W0afe 0z94fd430bfe6c5afe0K79ff0Z0 4fd220Ofe0m62ff5933ff0X0rfe 0I0Sfd0O27fe620□fd2e90fd74 46fd0h0kfe860ufd0□65fe326af e0F88fd0d8dff0V0Dff5f67fe0s 57ff648afd2426fd2369ff0v6dfd 0C0Ufd6000ff7a0Ffd0u50ff05 0Ife530Hfd0Q0Wfd0k56fd0B1 efe8a95ff0a0Xff0P0gfe0S5cfd2 70dff460zff910lfe020jfe0J0ife6 14efd410yfe038bfe0t03fe5d81f d4b72fe490Kfe040Tfe0p35fe7 90Pfd0G6bff920vfd2082fe7e4 5fd0ñ0Nfe0o53fe082aff0R0Eff 0n02fe063dff0L61fe4a06ff357f ff850Ñfe0j2dfe4f0Qfd070wfd0 Ñ0eff0r0pff830mfe0l32ff0f0Af d0i4ffe5877fe470hfd820qfe376 4fd0e0Mfd540tfd0y09fe0b0ffd*	79.5834	1.4397	10.7754	0.1118	0	(
Noisy random pseudo-hexadecimal (by substitution)	580K0▒410l0X350|0{90010s5 600&3d0r0U6e000□480b0{650 {0▒630&0□8f0"0930Off5d0^ff 5b0□ff550ffe230□ff720□fd730 nfe6f0Ufd910ufe4e0=ff570Xfd 7f0□fe440□fe840□fe510Hfe710 □ff5303fe5f0vff700□fe8c0□ff8a 09ff2a0□fe200hfd6b0□fd1e0□f e430:fe4d0off280Sff810Wff60 0.fd640gfd4207ff740%fd950,ff 860$fe590zff1f0Zfe780Mfe820 ?ff490kfd6d0,„fd940Efe5a0Gff3 e0iff250Nfd760_ff3b0…fe4f0 wfd890□ff210Œfe2f0cfd340ƒfe 8d0+fd2b0Ffe870Cfd2e0\fe3a 0#ff300<fe670`fe240Jfe960/fd 470]ff8b0fe770□fd260Vfe6605 fd500*fe5c0□fd 290Rff330[fe7a0@ff390– fd4a0'fe310†fd360pfe6108ff22 0Lfd460fd2d0efe7c0!ff2c0(fd8 30lfd370}fd690Bfe400Aff750;ff 4b0Yff5e0□fd800Dff2704fe320 2ff380Qfe680-fe 790qfd4c0)fd3c06ff620afe 880□fd6a0xfe7b0Tff7d0tfd450 >fd8e0jfe6c0sfd920yfd7e0Pfe8 50□fd520~fe540mff3f0dff | 66050▒590w0N800E0A860}0I 2d0□0410OO083f0{0]370m0A3 80t0▒900V0□680□0500Ufe53 0"ff8d0'fe300Jfd3e0vfe260`fe 360ofd290,fd220□fd6a0Pfd7a 0:ff960ufd270Lfe690@fd730/f f7c0□fd6707fd5a0Dfd820?ff42 0iff350bfd630fe250afe5103fd 3d0yfe3a02fd830□ff240[fd8c0 eff7f0xfe310_fe890□fe750\ff7 e0=fe6b0cfe8e0□fd440Qfe2b0 Œfe740‰fd280fe610hfd4 d0□fe5e0pff710`'fe450Xff400B fe6e01fd2e0`ff870□fd3b0dfe8 50zfe950nff1f0Sff340*ff390%f d6d0]fd760Kff5f0– fd520#fd430□fe940+ff1e0„fe4 b0sfe2c0-fe2a0Hff4e0M fd460Cfd8b0<ff840Tfe790lfe9 30kfe720fff910^fd7d0$fe620r ff770~ff700Gfd2104fd540□fd3 209fe5b0Yfd6f08fe600'fe810 Wfd8a0Zfe4f0jff640ff230…ff5 70Nff470†ff3300ff880|fd560Š fd5d0□fd200‹fd580&fe7b0Rfd 650Afe3c0Fe6c0lfe5c0gfd780 ;fe550{fe490,ff9206fe8f0)fe2f 0.fd4a0!fd480qfd4c0>fd	99.8243	1.2805	10.5764	0.0981	1	(
Noisy random pseudo-hexadecimal (by shifting)	5d0Z0▒860□0□8c0□0{60050w 360q0*2e0l08d0~0m30080{71 0Q0▒260+0□870n05606ff430 bff760fe7e01fd550- fd670`fd2f0<fd280gfe4b0Hff6c 0wfd320Off350:fe230Rff7d0"ff 290ofd8e0□fd480'fd9100fd4e0 @fe5a03fe5f0dfe200afe960zfe 890□ff5e0efe310tfe900†fe620 4fe7202ff930f f7c0'ff4a0rff790Yfd6e0vfe7f0T ff1e0#fd730‹fd8a0Œfe820□fd7 80Afe660Bff700□fe4c0ufd590/ ff740Šfe1f0^ff750\fd880,„fd3c 0(fe520?fd210kfd470|fe3a0Pff 3b0!ff8409fe5b0,fd490Ffd640 ‡fd830Jff8f0•ff390Lfd400]fd41 0=fe240Kfd2a0lfe570`ff800ffe 330Efd2b0%fd2c0.fe680_ff810 □fd420ife3f0`fd690Sff940– fd2d0hfe340Gff6f0fe920,fd220 □fe6b0$ff370&ff7b0jfe460cff6 d0mff530;ff4d0Wff3d07fe380} fd8b0Nfe450>fd540yfd440Mff 5c0xfe950pff850sfe250Vfd510 [fe6a0‰ff770□fe580Xfd630*fd 7a0"fd500)ff610…ff270Cfe3e0	2b0c0▒5e0x09930W0Z4b0@0 {4d0□0S510□0$880d0□7a0t0Z 810H0▒470s0□780!0610Lfd8 f07fe650Zff4a01fe3f0Gfd4f0□f d5f0#ff660□ff5a0bff7603fe80 05ff820afe690□fd940Mff920if e6f0$fe300ffe7f0;fd750_fe95 0<fd6a0Xfe620ff480□ff7e0/fe 5c08fe2204fe3d0□fd540Fff67 0□fd2e0(fe3700fd260Jfd200nf d2a0Eff350‰ff580Uff310-fe 5b0hfe500^ff910– ff250Yfe410)fe630Sff1e0=ff84 0□ff270Tfd360Šfe600Bfe8e0p fd830jfe320Dfd8d0rff860?fe6 40zfe8c0&ff960+fd380□ff7c0* fe420Vff6c0{fe460wfe3e02fe8 70eff7d0mff7b0%fe490Kff230 6fd680□fe450`fd2d0Afd340'fe 400}fd5709ff6b0Qfe560•fd29 0]fd530qfd520□ff700,ff4c0^fe 720ff790lff890~ff6e0[fd330gf e590†ff240.ff550vfe3c0ufd3b 0Pfd4e0/ff710Ife900kff390,ff 1f0Ofd3a0yfe430ff850‡fd6d0 ‘'fd740\fd730"ff440…fe8a0|f e280Rfe8b0off2f0Nfe770Cfd2	59.8281	0.6606	7.5558	0.1666	0	(

	Dff4f0Ufd650{fe	10ʿfd5d0>fe2c0:fe					
Noisy random pseudo-hexadecimal II (versión 2)	*63050▒880307750H0† 2a0‹ 0 =440605400T0'2f0;0“ 560S0† 5b0 0▒8e0@0□940d04e 0Œfd4f0>ff590gfe9300fe610rfd 1e0_fe1f0Š fe6f0'fd3c0 • fd5707 fd5d0Bff8a0uff2204fe4a0?fd29 0Rfe910Aff830tfd500\fe710, ff 490Ife890wff760Kfe740"fd320 • ff7d0f fe8c0ofe380Xff660 • fd 330bfd620%ff340''fd900(ff310 sff720Lfe860!ff6a0Ufd680|fe65 0†fd8b0-fe580cfd4b0+ff810 2fe6 d0“fd800‰ff390…fd3f0Zfe6c0 =fd6708fd3b0'fd950pff480Jff3 60Ž ff5a0jfe4201fe8d09fd3a0Q fd260xfd780fff6e0eff2e0iff540E fd460,fe770Pff300Vff7a0•ff700 Njd730Gfe3d0ˆff8f0lfd520vfd5c 0ff5e0*ff210 • fd230)ff530Ofe2 b0qfd820zfe470 • ff270hfd7e0D ff550Yff450}ff2d0Cff840'fd350 [fe4d0<fe3e0,,fd4c0^fe5f0mfd6 90Ffe410`fe2c0&fe7b0Mfd240 afe640□fe280‡fe510.fd7c0kff4 30nfd6b0{fe850/ff790$ff250#ff 870fe920yfd200Wff600:ff370]f e7f0~ff* | *950□0▒820z0□260t0□7b0#0 W300b01350‡ 0Œ930“ 0 830 • 0□940`0▒7a0c0□580‹ 03b0{ fe650, ff6f0Œfd520ƒ fd910vff4 f0nfd880_ff660+fd860Lfe250Y fe4b08fe710!ff5e0aff750Jff6c0 Wff6b0Zfd480• fd3a0$fe8e0' ƒ d390fd6e0Mfd500Hfe330gfe60 0ffe3806ff280Pff530=fd200Xf e470&fd420*ff560Ž fd3e0"ff2 a0Fff440Ofe8c0>fe210lfd450 „fe270xff370'fe2d0Qfe2c00fd5 10Rfd410Šfe8b0Tfe8d0sff460A fd2b0off2404fd800.fd6d0 ff5b0Efe760'fd3d0-fd2e0 wfd6407ff810 • ff430Ufd1e02fe 740qfd920†ff8a05ff400dfe850 <fe690jff7e0Cfe3109fe6301ff2 90ˇff590Kfd490 • fe3c03fd5c0| ff770Nff570''fe230Bfd7f0mfd7 90Gff4a0:fd5f0[ff4e0?fd780]fe 540@fe3f0…ff890pfe360ifd84 0%fd5a0)fd5d0~ff340;fd1f0Dff 4c0kfe870 • fd7c0ufe550^ff670 ,fd4d0/fd8f0rfd700hfd720\fe61 0¦ff680efe6a0(fe2f0Iff320‰fe 7d0fe620Sfe900€ fd730yfd220 Vfe*	111.2898	2.4496	12.4778	0.1960	0	(
Average			**545.94**	**1078.30**	**38.75**	**100.19**	**0.23**	(

In experiments, the *global average* has been computed with the modified *cross-validation* model for each different *encryption method*, separately (Rangel et al. 2025). Regarding the application of the novel *cross-validation* method, just one variant to the procedure, it has been carried out, which it is described as follows. For each encryption proposal: (1) First, the *encryption/decryption process* over five samples, it has been employed with size of 20%. (2) One *encryption/decryption* sample, it was sequentially extracted without replacement. The decryption process has been applied over four samples that they correspond to the 80% of encryption dataset. (3) Second step, it was repeated five times. After processing all the samples for each *encryption strategy*, the *average results, they* were computed, separately. This information in *Table 1*, it is shown. The columns *A* and *B*, they contain the results obtained with the employment of the *global average* and the *standard deviation*, respectively (the *times of encryption* and *decryption*, they were measured in milliseconds). In this work, the iterative procedure of the experiments, it was stopped when five repetitions of *ciphertext*, they were evaluated for each *encryption proposal* here presented (Rangel et al. 2025).

The best balanced results, they were obtained when the *DES* algorithm, it was employed. However, this alternative is not a good indicator because it has shown that it can produce decryption errors. Similarly, it occurs with the *noisy random pseudo-hexadecimal by substitution* implementation. On the other hand, the *symmetric AES-256* and *asymmetric ECIES-SECP256R1* alternatives, they have revealed good balance for safety of digital data, while *asymmetric RSA-2048* alternative, it has behaved slower than *standard encryption algorithms*, but it was faster than *noisy random pseudo-hexadecimal* alternatives. However, we must remember that *RSA* and *ECC* alternatives, they have been threatened by *quantum computing* advent. Regarding the comparison results of *noisy random pseudo-hexadecimal* alternatives. The best balanced results, they are always obtained between *25%* and *46%* when the *noisy random pseudo-hexadecimal by shifting* alternative, it was employed. Moreover, there exists very little decryption compatibility between standard version and the novel three proposed implementations. The *noisy random pseudo-hexadecimal* standard version, it could only be decryption by two *pseudo-hexadecimal* alternatives. Therefore, rest of the variants, they can be decryption by each proposal at *100%* with exception of the standard version. Thus, almost the *noisy random pseudo-hexadecimal by shifting* implementation, it shows itself that it is a good resource

to conduct the experimentation procedures, although other factors must be still analyzed (Rangel and Rangel 2024a; Rangel et al. 2025). In *Table 1*, this effect has been corroborated for the computed *global average*.

Tab. 2. In this table, the results of compatibility between noisy random pseudo-hexadecimal standard version and its three modifications based on substitution, shifting, and version 2, they are here shown when the "▒Welcome▒□" plaintext, it has been employed.

Encryption method	**Decryption compatibility between noisy random pseudo-hexadecimal alternatives**				
	Noisy random pseudo-hexadecimal I	Noisy random pseudo-hexadecimal (by substitution)	Noisy random pseudo-hexadecimal (by shifting)	Noisy random pseudo-hexadecimal II (versión 2)	"Noised" random pseudo-hexadecimal GAs (Rangel et al. 2024)
Noisy random pseudo-hexadecimal I	100%	70%	0%	100%	100%
Noisy random pseudo-hexadecimal (by substitution)	30%	100%	100%	100%	100%
Noisy random pseudo-hexadecimal (by shifting)	30%	100%	100%	100%	100%
Noisy random pseudo-hexadecimal II (versión 2)	30%	100%	100%	100%	100%
"Noised" random pseudo-hexadecimal GAs (Rangel et al. 2024)	30%	100%	100%	100%	100%
Average	**30%~44%**	**92%~94%**	**75%~80%**	**100%**	**100%**

The same can be stated about the results obtained. When processing only techniques based on *pseudo-hexadecimal* alternatives, it occurs because too long *ciphertext* is obtained. Therefore, some cases the execution of times, they are bigger than the other proposals here presented. This situation can achieve that the *cybercriminals*, they delay too much in the decryption process. Moreover, they should have known each location that it contains the *pseudo-hexadecimal noisy injection*. These tasks of data discovering might be very difficult. Besides, these schemes, they have always presented a *dynamical ciphertext* results. Repeated application of *dynamic encryption methods* can help us obtain distinct and better results in comparison with the *standard algorithms*. In this work, the iterative procedure of the experiments, it was stopped when five repetitions of *ciphertext*, they were produced with *noisy injection* strategies. In this research, it can observe in *Table 1*, several *static ciphertext* results which they have been obtained by *DES*, *3DES*, and *AES-256* algorithms. However, it does not mean, in all cases that they are vulnerable or insecure schemes. Instead, this situation is not presented (see *Table 1*), when the *noisy random pseudo-hexadecimal* alternatives, they were applied over *plaintext* input data. Therefore, all cases *dynamic ciphertext*, they were here shown. In case of the *standard encryption algorithms*, they were executed several experiments, where it was observed that some of them, suffers not only from the data security problem (with exception of the *AES*, *RSA*, and *ECIES*, proposals), but also the *encryption times*, they are too short in comparison with the *pseudo-hexadecimal* alternatives (see *Table 1*). The *"Noised" random pseudo-hexadecimal GAs* alternative (Rangel et al. 2024), it has shown to be slower than the other traditional proposals here evaluated when *plaintext*, it has been employed. Hence, these schemes have reported decryption errors.

However, it can produce vulnerability to the data security in the organisations and it might be very good for the *cybercriminal decryption strategies* (Rangel and Rangel 2024a). However, the *pseudo-hexadecimal by shifting* alternative, it has shown to be faster than the rest of proposals based on *pseudo-hexadecimal* encoding which they were here experimented. In these terms, the increased size of *ciphertext* with *pseudo-hexadecimal* encoding schema, they can be considered very good recommendation for confusing the *cybercriminals*. This schema does not put in risk data security of the information because the *ciphertext* result, it has included *noisy injection*. Nevertheless, additional disk storage space, it should be sacrificed to implement the proposals here recommended. Moreover, these alternatives based on *noisy injection*, it can become in very confused task. Due to that *cybercriminals* would have to guess practically where the *noisy injection*, it is encountered in the *ciphertext*. It would be a very complex task, even for *quantum computing*. Hence, these schemes can help obtaining a *dynamical ciphertext*, without to put in risk the data security of the organisations. Although these alternatives, they can present low risk for the *theft of digital data* because some noisy bigger proportion, it is inserted to the *ciphertext* (Rangel and Rangel 2024a; Rangel et al. 2025). Regarding the improvement of *encryption times*, they were obtained only with the *AES*, *ECIES*, and *Noisy random pseudo-hexadecimal by shifting*, alternatives.

Despite the successful results, a problem common to all these techniques is that they do not permit to control the selected maximum value of *random ordinals* of the *ASCII* table. Hence, can occur that some obtained random values, they will be out of range and therefore the *ASCII* can become as another encoding characters (e.g., *UTF-8*). In this work, this situation was very few observed due to the *mod 120* that it has been employed because it produces *ciphertext* sequences with allowed values in range of the *ASCII* table. However, two selected *plaintext* in our experiments, they contain the ' ▒' and ' □ ' characters with *9619* and *65533 ordinal* values, respectively. They are out of range of the *ASCII* table which they have not been accepted with some *standard encryption algorithms (e.g., DES* alternative). This situation could not be controlled since data input. Therefore, if we observe in *Table 1*, only *standard encryption algorithms*, they could hide the problem. On the other hand, a modification of the *ciphertext* by *dynamical alternatives* that they were here presented, it might be of interest. In particular, if it combines the simultaneous *random noisy pseudo-hexadecimal* schema with *artificial intelligence* based on *nearest neighbor rule* (Cover and Hart 1967; Rangel and Rangel 2024b), as well as, the employment of *mutation methodology* (Rangel and Rangel 2024a), it might be used for *camouflaging ciphertext*. Of course, this involves a great variety of possibilities that we will cover in the next future work.

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. There exists several studies that they have been involved with the problems produced by the presence of the vulnerable encryption schemes. If a method is well known, it will become in inadequate strategy. Hence, some authors novel *dynamical encryption* alternatives, they have suggested for handling to the *theft of digital data* problem. Therefore, this paper presents four *dynamic encryption* alternatives of *pseudo-hexadecimal* methodology based on *artificial intelligence* because it employs random and *heuristic methods* for selecting *encryption/decryption alphabeths* with *noisy random pseudo-hexadecimal* injection. Thereby, it can be employed in those real applications of the organisations. In the same way, the *noisy random pseudo-hexadecimal* for *noisy injection*, it offers an important contribution to amend deficiencies of the standard *encryption strategies* and to increase its usefulness. Experimental results with these *dynamic encryption* schemes, they have revealed that can cope with data security problems with high levels of trust. These proposals, in all cases, they allow the dynamical generalized results even better than those obtained with the *standard encryption algorithms* (see *Table 1*). We intend to do further research on this issue. One of the techniques that we are going to explore is the employment of some measures for increasing the *noisy injection* to the ciphertext based on the *nearest neighbor rule* and *pseudo-hexadecimal* proposals, as well as, they employment of *mutation methodology*. Of course, it involves a great variety of possibilities that we will cover in the next future work.

ACKNOWLEDGMENTS

This work was partially supported by Tecnológico Nacional de México (I.T. Ciudad Altamirano) as future work of a last project identified by: 19329.24-P.

REFERENCES

1. Rangel, E., Rangel, KU., and González, L. 2024. "Cifrado De Datos Dinámico Con Inteligencia Artificial, Utilizando El Nuevo Formato Pseudo-Hexadecimal". Revista Electrónica de Divulgación de la Investigación del SABES (Revista de la universidad del SABES), vol. 28, edición diciembre 2024. ISSN: 2007-3542. Editada por Sistema Avanzado de Bachillerato y Educación Superior en el Estado de Guanajuato. México. Available at: https://sabes.edu.mx/revista-electronica/27/# , (17/12/2024).
2. Rangel, E., and Rangel, KU. 2024a. "Novel Random Encryption Methods Based On Mutation Strategies Of Artificial Intelligence". Scientific and Practical Cyber Security Journal (SPCSJ), Vol. 8, No. 3. September Issue 2024. ISSN: 2587-4667 (electronic). Edited by SCSA - SPCSJ - BOAI. Published by Scientific Cyber Security Association in Georgia. Tbilisi, Georgia. Available at: https://journal.scsa.ge/issue/september-2024/, (05/11/2024).
3. Drzani, M. 2014. "Securing E-learning platforms". 2014 International Conference on Web and Open Access to Learning (ICWOAL), Dubai, United Arab Emirates, 1-4. Available at: https://doi.org/10.1109/ICWOAL.2014.7009237, (23/03/2025).
4. Mamman, J., Onoja, O., and Blessing, E. 2024. "Mitigating The Impact Of Phishing Attacks On The E-learning Infrastructure". Scientific and Practical Cyber Security Journal (SPCSJ), 8(3): 21-34. ISSN: 2587-4667. Scientific Cyber Security Association (SCSA). Tbilisi, Georgia. Available at: https://journal.scsa.ge/issues-archive/, (23/03/2025).
5. Rangel, E., Rangel, KU., Medrano, J., Bernal, CA., and González, L. 2023. "Algoritmo Genético Para Cifrado De Datos, Basado En Un Nuevo Concepto Pseudo-Hexadecimal Con Inteligencia Artificial". Tecnológico Nacional De México, Instituto Tecnológico de Cd. Altamirano. Sexto (VI) Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Noviembre, 2023. Cd. Altamirano, Estado De Guerrero, México. Available at: https://www.cdaltamirano.tecnm.mx/index.php/17-vi-congreso-nacional-de-investigacion-en-ciencia-e-innovacion-de-tecnologias-productivas/140-tecnm-40 , (23/03/2024).
6. Rangel, E., Rangel, KU., and González, L. 2025. "Dynamic Encryption Methods Based On Noisy Injection And Camouflaging Ciphertext Strategies With Artificial Intelligence", SPCSJ: Scientific and Practical Cyber Security Journal (Published by Scientific Cyber Security Association in Tbilisi, Georgia), ISSN: 2587–4667, vol. 9, no. 1, pp. 82-104, march issue. Available at: https://journal.scsa.ge/papers/dynamic-encryption-methods-based-on-noisy-injection-and-camouflaging-ciphertext-strategies-with-artificial-intelligence/, (23/04/2025).
7. Alvarez, D. 2019. "Algunos Aspectos Jurídicos Del Cifrado De Comunicaciones". Derecho PUCP, núm. 83, 2019, pp. 241-264. Pontificia Universidad Católica del Perú (2019). DOI: https://doi.org/10.18800/derechopucp.201902.008. Available at: http://www.redalyc.org/articulo.oa?id=533662765008, (23/03/2024).
8. Barandela, R., Sánchez, JS., García, V., and Rangel, E. 2003. "Strategies for Learning in Class Imbalance Problems". Pattern Recognition, Vol. 36, No. 3, pp. 849-851. Rapid and Brief Comunication (Pergamon) ISBN: (PII: S0031-3203(02)00257-1. 0031-3203/02/). Available at: https://doi.org/10.1016/S0031-3203(02)00257-1 , (23/03/2024).
9. Barranco, F., and Galindo, C. 2022. "Criptografía básica y algunas aplicaciones". Universidad Jaume I, Departamento de Matemáticas, Castellón, España. Available at: https://repositori.uji.es/xmlui/bitstream/handle/10234/201359/TFM_2022_Barranco_Bl%C3%A1zquez_FranciscoMiguel.pdf?sequence=1 , (23/03/2024).
10. Cover, TM., and Hart, PE. 1967. "Nearest Neighbor Pattern Classification". IEEE Transactions on Information Theory, Volume IT-13, January 1967, pages 21-27. Available at: https://ieeexplore.ieee.org/abstract/document/1053964/ , (23/03/2024).
11. Gómez, S. , Arias, JD., and Agudelo, D. 2012. "Cripto-Análisis Sobre Métodos Clásicos De Cifrado". Scientia Et Technica, vol. XVII, núm. 50, abril, pp. 97-102. Universidad Tecnológica de Pereira Pereira, Colombia. ISSN 0122-1701 97. Available at: http://www.redalyc.org/articulo.oa?id=84923878015, (23/03/2024).

12. Javidi, B., Zhang, GS., and Li, J. 1997. "Encrypted Optical Memory Using Double-random Phase Encoding". Appl. Opt. 36, 1054-1058. Available at: https://pubmed.ncbi.nlm.nih.gov/18250772/ , (23/03/2024).
13. Rangel, E. 2022. "La Regla De Los k Vecinos Más Cercanos (k-NN) Basada En Distancia De Manhattan (City-Block) Para Mejorar La Clasificación De Patrones". Publicado En: Quinto Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Tecnológico Nacional De México (campus: Instituto Tecnológico de Cd. Altamirano). Noviembre, 2022. Cd. Altamirano, Estado De Guerrero, México. Available at: http://erangel.coolpage.biz/pappers/edgarrangel2022.pdf , (23/03/2024).
14. Reddaiah, BA. 2019. "Study on Genetic Algorithms for Cryptography". International Journal of Computer Applications (0975 – 8887). Volume 177 - No. 28, December 2019. Department of Computer Applications. Yogi Vemana University Kadapa, A.P, India. Available at: https://www.researchgate.net/publication/338012809_A_Study_on_Genetic_Algorithms_for_Cryptography , (23/03/2024).
15. Sebas, C. 2023. "¿Qué son los Algoritmos Genéticos en las Inteligencias Artificiales?". Manuales y Tutoriales de Informatica. Available at: https://aprendeinformaticas.com/ia/ ,(23/03/2024).
16. Ennu, G., Martes, M., and Boratto, L. 2018. "A multi-brometne system for continuous student authentication in e-learning platforms". Pattern Recognition Letters, 113, 83-92. Available at: https://doi.org/10.1016/j.patrec.2017.03.027, (23/03/2025).
17. Paul, S., Dasgupta, P., Naskar, PK., and Chaudhuri, A. 2017. "Secured image encryption scheme based on DNA encoding and chaotic map", Review Of Computer Engineering Studies, IIETA: International Information and Engineering Technology Association. ISSN: 2369-0755, e-ISSN: 2369-0763, vol. 4, no. 2, pp. 70-75, issue june, doi: 10.18280/rces.040206. Available at: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://iieta.org/sites/default/files/Journals/RCES/04.02_06.pdf&ved=2ahUKEwjVlYfordOMAxVVLUQIHXknBWsQFnoECB4QAQ&usg=AOvVaw3yvem4PsuKbMM9009owuAJ, (12/04/2015).
18. Delman, B. 2004. "Genetic Algorithms in Cryptography". Thesis for the Degree of Master of Science in Computer Engineering. Rochester Institute of Technology, RIT Scholar Works. Department of Computer Engineering. Available at: https://scholar.google.com.mx/scholar_url?url=https://repository.rit.edu/cgi/viewcontent.cgi%3Farticle%3D6460%26context%3Dtheses&hl=es&sa=X&ei=cbaZZoadNt246rQPnd604AU&scisig=AFWwaeaMfCM5ORUFQN6DU4LA3aEG&oi=scholarr, (23/03/2024).
19. Granados, G. 2006. "Introducción A La Criptografía", UNAM (Universidad Nacional Autónoma de México), Revista Digital Universitaria (ISSN: 1607 - 6079), vol. 7, no. 7, julio. Available at: http://www.revista.unam.mx/vol.7/num7/art55/int55.htm, (12/04/2025).
20. Kalsi, S., Kaur, H., and Chang, V. 2018. "DNA Cryptography and Deep Learning using Genetic Algorithm with NW algorithm for Key Generation". Convergence of Deep Machine Learning and Nature Inspired Computing Paradigms for Medical Informatics. Image & Signal Processing. In Journal of Medical Systems, volume 42. Article number: 17. DOI: https://doi.org/10.1007/s10916-017-0851, (06/06/2025).
21. Mendoza, JC. 2008. "Demostración De Cifrado Simetrico Y Asimetrico". Ingenius. Revista de Ciencia y Tecnología, núm. 3, pp. 46-53. Universidad Politécnica Salesian. Cuenca, Ecuador. ISSN: 1390-650X. Available at: http://www.redalyc.org/articulo.oa?id=505554806007, (23/03/2024).
22. Rueda, JE., Romero, AL., and Castro, LM. 2005. "Criptografía Digital Basada En Tecnología Óptica". Bistua: Revista de la Facultad de Ciencias Básicas, vol. 3, núm. 2, julio 2005, pp. 19-25. ISSN: 0120 - 4211. Universidad de Pamplona, Colombia. Available at: http://www.redalyc.org/articulo.oa?id=90330203, (23/03/2024).
23. Singh, S. 2000. "Los códigos secretos", Madrid: Debate (ISBN: 9788483062784), Madrid, España. Available at: https://scholar.google.com.mx/scholar?hl=es&as_sdt=0%2C5&q=Los+c%C3%B3digos+secretos&btnG=#d=gs_qabs&t=1744657839049&u=%23p%3DB4VAJ9oiNSQJ, (14/04/2025).
24. Susilo, W., Tonien, J., and.Yang, G. 2021. "Divide and capture: An improved cryptanalysis of the encryption standard algorithm RSA", Computer Standards & Interfaces (ISSN: 0920-5489), vol. 74 (1): 103470, february, doi: https://doi.org/10.1016/j.csi.2020.103470. Available at: https://www.sciencedirect.com/science/article/pii/S0920548920303561, (06/06/2025).
25. Baklaga, L. 2024. "Leading The Way In Quantum-Resistant Cryptography For Everyday Safety". Scientific and Practical Cyber Security Journal (SPCSJ), 8(3): 65-73. ISSN: 2587-4667. Scientific Cyber Security Association (SCSA).Tbilisi, Georgia. Available at:

https://journal.scsa.ge/papers/leading-the-way-in-quantum-resistant-cryptography-for-everyday-safety/, (23/03/2025).

26. Bavdekar, R., Eashan-Jayant C., Ankit A., and Tiwari, K. 2023. "Post Quantum Cryptography: A Review of Techniques, Challenges, and Standardizations." In 2023 International Conference on Information Networking (ICOIN).
27. Dhany, HW., Izhari, F., Fahmi, H.,Tulus, M., and Sutarman, M. 2018. "Encryption and Decryption using Password Based Encryption, MD5, and DES". Published by Atlantis Press. ISSN: 2352-5398. Available at: http://creativecommons.org/licenses/by-nc/4.0/, (23/03/2024).
28. Kumar, A., and Sharma, S. 2021. "A Study on Historical Cryptographic Techniques: Caesar Cipher to DES". International Journal of Advanced Science and Technology, 30(2), 555-564.
29. Progress Software Corporation, Telerik. 2022. "Cifrado Y Transferencia De Archivos: Los Mejores Cifrados Seguros Para La Transferencia De Archivos". Ipswitch Blogs (2020-2022). Access/Revisión: 31-08-2024. Available at: https://ipswitch.com/amp/es/los-mejores-cifrados-seguros-para-la-transferencia-de-archivos/, (23/03/2024).
30. Rodríguez, J. 2020. "Operadores Genéticos Aplicados A La Criptografía Simétrica". Proyecto De Grado. Universidad Distrital Francisco José De Caldas. Facultad De Ingeniería. Ingeniería De Sistemas. Bogotá, Colombia. Available at: https://repository.udistrital.edu.co/handle/11349/28192 , (23/03/2024).
31. Van, HC., and Jajodia, S. 2011. "Encyclopedia Of Cryptography And Security". Springer Science & Business Media, 2011. 1416p. ISBN: 978-14419-5907-2.
32. Van-Tilborg, HCA. 2005. "Encyclopedia Of Cryptography And Security". Springer, pp 114-115, 201-202. TUE Research portal. Available at: https://doi.org/10.1007/0-387-23483-7, (09/11/2024).
33. Rahman, M.S., and Hossain, M.S. 2021. "A Secure Private Key Cryptography Scheme Using RSA and AES". Journal of Cybersecurity, 1(1), 1-9.
34. Singh, P., and Kumar, P. 2022. "Optical encryption using fractional Fourier transform and spiral phase plate". Optics Communications, 503, 127629.
35. Daemen, J., and Rijmen, V. 2002. "The Design of Rijndael: AES - The Advanced Encryption Standard". Springer. DOI:10.1007/978-3-662-04722-4, (23/03/2025).
36. NIST. 2001. "FIPS PUB 197: Advanced Encryption Standard (AES)". U.S. Department of Commerce.
37. Dang, QH., and Le, HQ. 2022. "Improved cryptanalysis of the RSA algorithm using side-channel attacks". Journal of Information Security and Applications, 65, 103313. 00.
38. Luciano, D., and Prichett, G. 1987. "Cryptology: From Caesar Ciphers To Public-key Cryptosystems". The College Mathematics Journal, vol 18 pp 2-17. Available at: http://www.jstor.org/stable/2686311 , (23/03/2024).
39. Hankerson, D., Hernandez, JL., and Menezes, AJ. 2004. "Software implementation of elliptic curve cryptography over binary fields". K. Ko¸c and C. Paar (Eds.): CHES 2000, LNCS 1965, pp. 1–24, 2000. Springer-Verlag Berlin Heidelberg 2000. Available at: https://idp.springer.com/authorize?response_type=cookie&client_id=springerlink&redirect_uri=https%3A%2F%2Flink.springer.com%2Fcontent%2Fpdf%2F10.1007%2F3-540-44499-8_1.pdf, (23/03/2025).
40. Montgomery, PL. 1987. "Speeding up the Pollard rho method". Mathematics of Computation, 48(177), 453-456.
41. Rivest, R. L., Shamir, A., and Adleman, L. 1978. "A method for obtaining digital signatures and public-key cryptosystems". Communications of the ACM, 21(2), 120-126. Available at: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://people.csail.mit.edu/rivest/Rsapaper.pdf&ved=2ahUKEwjg2fiM2qGMAxXbhu4BHe9iCSsQFnoECDUQAQ&usg=AOvVaw1FtI2T7P32pKAb6jnZSqxi, (23/03/2025).
42. Baker, M., and Schiller, J. 2015. "ECIES: Elliptic Curve Integrated Encryption Scheme". In Cryptography and Network Security, pp. 245-263. Springer. Available at: https://medium.com/asecuritysite-when-bob-met-alice/elliptic-curve-integrated-encryption-scheme-ecies-encrypting-using-elliptic-curves-dc8d0b87e aa, (23/03/2025).
43. NIST. 2013. "Special Publication 800-56A Revision 2: Recommended methods for key establishment using public key cryptography". NIST Special Publication 800-56A. Available at: (https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-56ar2.pdf&ved=2ahUKEwio1eCz3KGMAxWgJ0QIHYc4BXQQFnoECBsQAQ&usg=AOvVaw367-qADImRilvhabe1UtQr), (https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://csrc.nist.gov/pubs/sp/800/56/a/r2/final&ved=2ahUKEwio1eCz3KG

MAxWgJ0QIHYc4BXQQFnoECBwQAQ&usg=AOvVaw183JO5CLTxYuENINrMwqgL), (23/03/2025).

44. Iavich, M., Kuchukhidze, T., and Gagnidze, A. 2024. "Post-quantum Digital Signature Using Verkle Trees And Lattices". Scientific and Practical Cyber Security Journal (SPCSJ), 8(3): 35-52. ISSN: 2587-4667. Scientific Cyber Security Association (SCSA). Tbilisi, Georgia. Available at: https://journal.scsa.ge/issues-archive/, (23/03/2025).
45. Fuegner, P. 2024. "Are RSA and AES Both at Risk From the Quantum Threat?". QuSecure, Inc. Available at: https://www.qusecure.com/are-rsa-and-aes-both-at-risk-from-the-quantum-threat/#:~:text=The%20emergence%20of%20quantum%20computers,efficiently%20factoring%20large%20prime%20numbers, (2025-02-08).
46. Sengupta, S., and Ghosh, S. 2023. "Quantum Computing Encryption Threats: Why RSA and AES Are at Risk". Journal of Cryptographic Research.
47. Thakur, J., and Kumar, N. 2011. "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis". International Journal of Emerging Technology and Advanced Engineering, 6-12.
48. Iyengar, S. S., Kannan, R., and Ganapathi, S. 2021a. "Inteligencia artificial y criptografía: Tendencias y desafíos". IEEE Transactions on Emerging Topics in Computing, 9(2), 833-844.
49. Li, J., and Wang, X. 2022. "New Collision Attacks on SHA-1 Based on Differentiated Path Construction". Journal of Cryptology, 35(3), 1-25.
50. Singh, A.K., Kumar, P., and Singh, R. 2021. "Aplicación de la inteligencia artificial en la criptografía: Una revisión". Journal of Intelligent Information Systems, 67(2), 257-275.
51. Rangel, E. 2002. "Vecinos Envolventes para Variantes de la Regla del Vecino más Cercano". MSc.Thesis, Instituto Tecnológico de Toluca, México.
52. Hartmann, AK. 2020. "Heuristic search in graphs". Journal of Artificial Intelligence Research, 68, 1-33.
53. Russell, SJ., and Norvig, P. 2020. "Inteligencia artificial: Un enfoque moderno", (4ª ed.). Pearson.
54. Sánchez, JS., Pla, F., and Ferri, FJ. 1997. "Prototype selection for the nearest neighbor rule through proximity graphs". Pattern Recognition Letters 18, 507-513.
55. Mitchell, TM. 2020. "Machine learning", (2ª ed.). McGraw-Hill.
56. Ross-Quinlan, J. 1993. "C4.5: Programs for Machine Learning". Morgan Kaufmann, San Mateo, CA.
57. Kuncheva, LI., and Jain, LC. 1999. "Nearest Neighbor Classifier: Simultaneous editing and feature selection". Pattern Recognition Letters, 20, 1149-1156.
58. Murphy, KP. 2022. "Probabilistic machine learning: An introduction". MIT Press.
59. Reddaiah, B. 2016. "A Study on Pairing Functions for Cryptography". IJCA (0975-8887), Vol. 149, No. 10, September, pp.4-7.
60. Skalak, DB. 1994. "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms". In: Proceedings of the Eleventh International Conference on Machine Learning (ML94). Morgan Kaufmann, pp. 293-301.
61. Bishop, C.M. 2022. "Pattern recognition and machine learning", (2ª ed.). Springer.
62. Kanal, L.N. 1963. "Statical methods for pattern classification". Philco Rept, originally appeared in T. Harley et al., Semi-automatic imagery screening research study and experimental investigation, Philco Reports, V043-2 and v043-3, Vol. I, sec. 6 and Appendix H, prepared for U.S. Army Electronics Research and Development Lab. Under Contract DA-36-039-sc-90742, March 29.
63. Johns, M.V. 1961. "An empirical Bayes approach to non-parametric two-way classification". In Studies in Item Analysis and Prediction, H. Solomon, Ed Stanford, Calif,: Standford University Press.
64. Sebestyen, G. 1962. "Decision Making". Process in Pattern Recognition. New York: Macmillan, pp. 90-92.
65. Chen, X., and Wang, J. 2022. "Quantum machine learning: A survey". Information Sciences, 593, 257-275.
66. Goodfellow, I., Bengio, Y., and Courville, A. 2021. "Deep learning". MIT Press.
67. Nils-Nilson. 1965. "Learning Machines". New York: McGraw Hill, pp 120-121.
68. Han, J., Pei, J., and Kamber, M. 2022. "Data mining: Concepts and techniques", (4ª ed.). Morgan Kaufmann.
69. Hand, D., Mannila, H., and Smyth, P. 2001. "Principles of Data Mining". The MIT Press.
70. Tan, P.N., Steinbach, M., and Kumar, V. 2022. "Introduction to data mining", (2ª ed.). Pearson.
71. Zhang, Y., and Yang, Q. 2022. "Deep learning for data mining: A survey". ACM Computing Surveys, 54(2), 1-37.
72. Goodfellow, I., Bengio, Y., & Courville, A. 2016. "Deep learning". MIT Press.

73. Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. "ImageNet classification with deep convolutional neural networks". Advances in Neural Information Processing Systems, 25.
74. LeCun, Y., Bengio, Y., and Hinton, G. 2015. "Deep learning". Nature, 521(7553), 436-444.
75. Clark, A. 1994. "Modern optimisation algorithms for cryptanalysis". In Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, November 29 December 2, (pp. 258-262).
76. Griindlingh, W., and Van-Vuuren, JH. 2002. "Using Genetic Algorithms to Break a Simple Cryptographic Cipher". Retrieved March 31, 2003 from http://dip.sun.ac.za/~vuuren/abstracts/abstr_genetic.htm.
77. Matthews, RAJ. 1993. "The use of genetic algorithms in cryptanalysis". Cryptologia, 17(4), 187-201.
78. Bruzzone, L., and Serpico, S.B. 1997. "Classification of Imbalanced remote-sensing data by neural networks". Elsevier Science B.V. , 0167-8655, 97. PH S0167-8655 (97) 00109-8.
79. Rangel, E., and Rangel, KU. 2024b. "La Regla Del Vecino Más Cercano Como Alternativa Para Inyectar Ruido A Mensajes Encriptados Por El Algoritmo: Noised Random Hexadecimal". INTELETICA. Revista de Inteligencia Artificial, Ética y Sociedad. 1, 2 (Dec. 2024), 1–15. Editado por IBERAMIA (Iberoamerican Society of Artificial Intelligence: Sociedad Iberoamericana de Inteligencia Artificial). ISSN: 3020-7444. España. Available at: https://inteletica.iberamia.org/index.php/journal/article/view/16, (23/03/2025).
80. Bošnjak, L., Sres, J., and Brumen, B. 2018. "Brute-force and dictionary attack on hashed real-world passwords". Conference: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). DOI: 10.23919/MIPRO.2018.8400211. Available at: https://www.researchgate.net/publication/326700354_Brute-force_and_dictionary_attack_on_hashed_real-world_passwords, (23/05/2025).
81. Python.org. 2024. "The Python Network". Python.org. Available at: https://www.python.org/downloads/, (18/11/2024).
82. Pydroid3 versión 7.4_arm64. 2024. [IDE for Python 3. Lenguaje de programación y compilador]. Google Play Store. Available at: https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&pli=1, (18/11/2024).
83. PyCryptodome. 2025. "Crypto.Cipher package. Introduction". Available at: https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html, (30/03/2025).
84. PyPI. 2024. "Pycryptodome 3.21.0". Python Software Foundation. Available at: https://pypi.org/project/pycryptodome/, (13/12/2024).
85. ECIESPy. 2025. "ECIESPy 0.4.4". Python Software Foundation. Available at: https://pypi.org/project/eciespy/, (29/03/2025).