

NOVEL DYNAMIC CAST-128 ENCRYPTION SCHEME AND COMPARISON WITH FIVE RANDOM NOISY INJECTION STRATEGIES WITH ARTIFICIAL INTELLIGENCE

Rangel-Lugo, Edgar ¹, Rangel-Ríos, Kevin Uriel ², González-Vidales, Leonel ³, Bernal-Beltrán, Carlos Alberto ⁴, Ascencio-Antúnez, Lucero De Jesús ⁵, Reynoso-Andrés, Rosa Isabel ⁶ and Rodríguez-Torres, César Del Ángel ⁷

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Sistemas y Computación ¹

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Sistemas y Computación ²

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Sistemas y Computación ³

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Subdirección Académica ⁴

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Planeación, Prog. y Presupuestación ⁵

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Desarrollo Académico ⁶

Tecnológico Nacional de México. Instituto Tecnológico de Ciudad Altamirano. Depto. de Sistemas y Computación ⁷

ABSTRACT: The cyberattack's issue is receiving growing attention. The loss and theft of digital data, they are derived consequences of this type of problem. Digital data theft poses a significant risk to an organisation's financial well-being. Dynamic encryption methods can handle this situation effectively, due to that they produce distinct ciphertexts for the same unique plaintext inputs. In this paper, several aspects related to this subject, they are studied. Our aim is focused on cybersecurity strategies, which they use the *random noisy* schemes with artificial intelligence (AI). This new approach involves injecting noise into ciphertext. These alternatives are well known as *random noisy strategies*. This study is a future work on *random noisy* strategies because it was observed that they have not been experimented with the *CAST-128* encryption algorithm on ciphertext. This limitation is not ideal for organisations, particularly those that they have opted for the employment of *CAST-128* algorithm. Besides, in this work, experimental results have shown that standard *CAST-128* encryption algorithm, it can obtain dynamic ciphertext outputs. Hence, is presented here a new opportunity for organisations regarding the employment of the *CAST-128* based on *noisy injection* methodology. This novel definition, named here as the *random noisy CAST-128* strategy. It is introduced as a dynamic data encryption alternative. Finally, a comparison of results with five *random noisy strategies* based on *DES*, *3DES*, *AES-256*, *CAST-128*, and *Blowfish* algorithms, they are also presented because these strategies here recommended, they can be resistant to decryption by cybercriminals and even by quantum computing algorithms.

KEYWORDS: *Random noisy strategies, dynamic encryption methods, applications of AI, cybersecurity strategies.*

1. INTRODUCTION

A cybersecurity strategy (Delman 2004; Kalsi et al. 2018; Mendoza 2008), it is only as strong as its weakest link, because if one method is vulnerable to cyberattacks (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b, 2025c), it can lead to digital data theft (Rangel et al. 2023, 2025c), and significant financial losses for organisations (Rangel and Rangel 2024a, 2024b; Rangel et al. 2025a, 2025c; Rangel et al. 2024a). Most of these cases (Álvarez 2019; Barranco and Galindo 2022; Gómez et al. 2012; Javidi and Horner 1994; Reddaiah 2019; Sebas 2023), they 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 (Rangel and Rangel 2024a; Rangel et al. 2025a). Several approaches have been proposed to tackle the issue of digital data theft. These can be broadly categorised into three main strategies: Firstly, regularly updating cybersecurity strategies (Rangel and Rangel 2024a). Second, the employment of dynamic encryption methods (Rangel et al. 2025a). Third, the noisy injection application on ciphertext (Rangel et al. 2023, 2024a, 2025a, 2025b; Rangel and Rangel 2025a), which it has shown promise in certain practical applications. Our research centres on encryption techniques (Rangel et al. 2025c), based on noisy injection strategies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). According to (Rangel et al. 2025c), these techniques employ mathematical equations or statistical methods to convert plaintext (S_i) into ciphertext (C_i), ensuring that only authorised parties can access the information (Rangel et al. 2024a; Rangel and Rangel,

2024b). The S_i denotes the original data sequence, and the C_i is the encrypted result. Dynamic encryption is characterised by generating different results with the same plaintext input data, whereas static encryption produces consistent results (Rangel et al. 2025c). Encryption algorithms can be symmetric, utilising a single secret key, or asymmetric, using a private and public key pair (Rangel et al. 2025c). Data encryption involves converting plaintext into ciphertext, while the reverse process is known as data decryption (Gómez et al. 2012; Javidi and Horner 1994; Rangel et al. 2023, 2024a, 2025c; Reddaiah 2019; Sebas 2023; Paul et al. 2017). Asymmetric algorithms, such as *Rivest-Shamir-Adleman* or *RSA* (Mendoza 2008; Rangel et al. 2025a, 2025b, 2025c; Barranco and Galindo 2022; Gómez et al. 2012; Oppliger 2005; Stinson and Paterson 2019; Van-Tilborg 2005; Baklaga 2024; Bavdekar et al. 2023; Dang and Le 2022; Luciano and Prichett 1987; Rahman and Hossain 2021; Rodríguez 2020), the *Elliptic Curve Cryptography* or *ECC* (Rangel et al. 2025a, 2025b, 2025c; Oppliger 2005; Stinson and Paterson 2019; Van-Tilborg 2005; Baklaga 2024; Bavdekar et al. 2023; Baker and Schiller 2015; Hankerson et al. 2004; Montgomery 1987; NIST 2013), and *ElGamal* (Barranco and Galindo 2022; Oppliger 2005; Stinson and Paterson 2019; Van-Tilborg 2005), they utilise a pair of keys - private and public - for the encryption process. Dynamic encryption has shown promising results in recent research (Rangel et al. 2025a, 2025c; Rangel and Rangel, 2024b), when applied to asymmetric algorithms. Since asymmetric algorithms are not within the scope of this work, they are not examined here and they could be considered for future works. On the other hand, some examples of symmetric key cryptography algorithms include *DES*, *3DES* or *TripleDES*, *AES*, *Blowfish*, and *CAST*, among others. Additionally, several symmetric key cryptography algorithms, they are here employed, including the *Data Encryption Standard* or *DES* (Mendoza 2008; Barranco and Galindo 2022; Gómez et al. 2012; Baklaga 2024; Bavdekar et al. 2023; Rodríguez 2020; Dhany et al. 2018; Kumar and Sharma 2021), the *Triple Data Encryption Standard* or *3DES* (Gómez et al. 2012; Van-Tilborg 2005; Baklaga 2024; Bavdekar et al. 2023; Van and Jajodia 2011), *Blowfish* (Baklaga 2024; Bavdekar et al. 2023; Schneier 1994; Schneier 2000; AL-Maqtari and AL-Maqtari 2024; Muhammed et al. 2024), *CAST* of *Carlisle Adams & Stafford Tavares* (Wang et al. 2017), and the *Advanced Encryption Standard* or *AES* (Barranco and Galindo 2022; Van-Tilborg 2005; Baklaga 2024; Bavdekar et al. 2023; Rahman and Hossain 2021; Rodríguez 2020; Saini et al. 2023; Daemen and Rijmen 2002). In this research, the variants: *CAST-128* (Wang et al. 2017), and *AES-256* (Muhammed et al. 2024; Saini et al. 2023; Daemen and Rijmen 2002), they have been employed respectively. According to (Baklaga 2024), *AES* is a symmetric block cipher that operates with different block sizes and key lengths of 128, 192, and 256 bits are supported. In contrast, *DES* processes 64-bit blocks of plaintext to generate 64-bit blocks of ciphertext, using a combination of substitution and permutation across multiple rounds, with decryption carried out in reverse. However, 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 utilises a variable-length block cipher with key lengths between 32 and 448 bits (Baklaga 2024; Bavdekar et al. 2023). In most cases, *Blowfish* is implemented with a block size of 64 bits. Similarly, *CAST5* or *CAST-128* (Wang et al. 2017), it is a symmetric algorithm that is the predecessor to *CAST-256* and it is based on a fixed-length block cipher. According to (Wang et al. 2017), *CAST-128* exhibits good resistance to various forms of cryptanalysis. As a *DES*-like *SPN* cryptosystem (Wang et al. 2017), *CAST-128* operates with a 64-bit block size and key sizes ranging from 40 to 128 bits. Besides, *CAST-128* is typically used with a 64-bit block size, offering good performance features for data security, similar to its successor. Some studies (Rangel et al. 2025a, 2025c), they have found that symmetric encryption algorithms yield static ciphertext results. However, this does not mean that these algorithms are inherently insecure (Rangel et al. 2025a). Still, the rise of quantum computing poses a threat to some standard encryption methods (Rangel et al. 2025c; Baklaga 2024; Bavdekar et al. 2023; Iavich et al. 2024). In contrast, others authors (Fuegner 2024; Sengupta and Ghosh 2023; Thakur and Kumar 2011), they warn that the *AES* and *RSA* algorithms have been also threatened by quantum computing advent (Rangel et al. 2025c; Baklaga 2024; Iavich et al. 2024). In this paper, several aspects related to this subject are studied. Although that quantum computing is not used in this work, other measures have been proposed being they applied with symmetric algorithm variants (Rangel et al. 2025a). According to various researchers (Rangel and Rangel 2024a, 2024b, 2025a; Rangel et al. 2023, 2024a, 2025a, 2025b, 2025c), the dynamic encryption methods offer a promising approach to bolstering security by incorporating more

noise and redundancy into ciphertext outputs (Rangel et al. 2023, 2025a, 2025c). Additionally, employing noisy injection strategies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), it is considered a good practice for encryption. In the context of noisy injection, extra characters from ASCII or UTF-8 encoding, they are added to a plaintext or ciphertext, which they are not part of the original message content (Rangel et al. 2025c). A key hypothesis of this research is that these strategies can create confusion for cybercriminals, as indicated some studies (Rangel et al. 2025a, 2025c). The development of noisy injection strategies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), it often incorporates artificial intelligence (Rangel et al. 2024a, 2025a, 2025b, 2025c; Rangel and Rangel 2024b, 2025a; Rangel 2002, 2022), drawing from the principles of AI-based cryptography (Delman 2004; Kalsi et al. 2018; Mendoza 2008; Rangel and Rangel 2024a, 2024b; Rangel et al. 2024a, 2025c; Reddaiah 2019; Sebas 2023; Dang and Le 2022; Iyengar et al. 2021; Liu and Wang 2022; Singh et al. 2021). According to (Rangel et al. 2024a; Rangel 2002, 2022), they refer that the purpose of artificial intelligence (AI), it is to make machines think (Rangel et al. 2025c). In line with this, approaches like heuristic methods (Rangel et al. 2024a, 2025a, 2025c), they have been developed using predefined rules to solve problems. These can be used in structured models such as decision trees (Mitchell 2020; Ross-Quinlan 1993; Russell and Norvig 2020), and graphs (Russell and Norvig 2020; Hartmann 2020; Sánchez et al. 1997), to mention few. There exists also AI-driven alternatives that leverage random methods (Rangel et al. 2024a; Kuncheva and Jain 1999; Murphy 2022; Reddaiah 2016; Skalak 1994), which they include random number selection with or without replacement (Rangel et al. 2025c). These can be combined with models such as genetic algorithms (Delman 2004; Kalsi et al. 2018; Rangel et al. 2023, 2024a; Reddaiah 2019; Sebas 2023; Kuncheva and Jain 1999; Skalak 1994; Clark 1994; Griindlingh and Van-Vuuren 2003; Matthews 1993), and the Monte Carlo or MC1 algorithm (Skalak 1994), as well as some models based on artificial neural networks (Murphy 2022; Bruzzone and Serpico 1997; Goodfellow et al. 2021), among other. In these terms, various AI-based cryptography alternatives, they have been proposed by several authors (Delman 2004; Kalsi et al. 2018; Mendoza 2008; Rangel and Rangel 2024a, 2024b; Rangel et al. 2024a, 2025c; Gómez et al. 2012; Reddaiah 2019; Sebas 2023; Dang and Le 2022; Rodríguez 2020; Iyengar et al. 2021; Liu and Wang 2022; Singh et al. 2021; Clark 1994; Griindlingh and Van-Vuuren 2003; Matthews 1993). For example, Singh et al. (2021), they have presented a review of AI applications in cryptography. In contrast, Iyengar et al. (2021), they discuss trends and challenges in this field. Meanwhile, Mendoza (2008), explores both asymmetric and symmetric cryptography demonstrations. Additionally, Reddaiah (2016), refers over the pairing functions in AI-based cryptography through experimentation. Later, the same author (Reddaiah 2019), has conducted a groundbreaking study on genetic algorithms (GAs) for cryptography, with a focus on e-commerce applications. Besides, Dang and Le (2022), highlight an enhanced cryptanalysis of the RSA algorithm through side-channel attacks. On the other hand, Liu and Wang (2022), introduce a new collision attack strategy that it uses differentiated path construction. Following the common practice, there exists some studies (Delman 2004; Matthews 1993), based on genetic algorithms in cryptography. In this context, a modern optimisation algorithms for cryptanalysis, it was presented by Clark (1994). On the flip side, Griindlingh and Van-Vuuren (2003), have revealed that genetic algorithms can successfully break certain simple cryptographic ciphers. According to (Rodríguez 2020), refers on the application of genetic operators to symmetric cryptography using GAs. Moreover, Liu and Wang (2022), have introduced a quantum cryptanalysis technique for lattice-based protocols. Regarding the noisy injection strategies (Rangel and Rangel 2024a, 2024b, 2025a; Rangel et al. 2023, 2024a, 2025a, 2025b, 2025c), encompass multiple approaches that they can be classified into three distinct categories. The first category involves the use of *pseudo-hexadecimal* format. Notably, the '*Noised* random pseudo-hexadecimal GAs methodology has been described in other studies (Rangel et al. 2023, 2024a). It is a schema based on genetic algorithm that it was presented as a dynamic encryption alternative. Due to that *pseudo-hexadecimal* GAs, it has reported disadvantages, Rangel et al. (2024a), have presented a successor which it was named as "*Noised*" random pseudo-hexadecimal (without GAs). Subsequently, Rangel et al. (2025c), have introduced four dynamic alternatives based on the *pseudo-hexadecimal* scheme, referred to as "*noisy random pseudo-hexadecimal*" strategies. These strategies aim to inject noise into ASCII characters when a new *pseudo-hexadecimal* format has been employed for confusing to the cybercriminals. These schemes have been applied exclusively to plaintext. Second approach, it involves combining AI-based noisy injection with the 1-NN rule, as

discussed by other authors (Rangel 2002, 2022; Barandela et al. 2003; Lewis and Catlett 1994; Cover and Hart 1967). In this framework, Rangel and Rangel (2024b), they have discussed the "Noised" *random 1-NN with hexadecimal* encoding based on AI. Likewise, Rangel and Rangel (2024c), so they examined the integration of "Noised" *random pseudo-hexadecimal* format with 1-NN. Both methodologies have demonstrated that these schemes can enhance the security of digital data through double noisy injection over ciphertext outputs, albeit at the cost of increased disk storage space. These strategies were additionally applied to plaintext. This paper did not investigate the *pseudo-hexadecimal* methodology, but it could be a potential area for future work. The third method leverages *random Caesar II mod 120* (Rangel and Rangel 2024a, 2024b, 2025a; Rangel et al. 2025a; 2025b), for noisy injection, being applied to plaintext (Rangel and Rangel 2024a, 2024b; Rangel et al. 2025a), and ciphertext (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), being obtained via standard encryption. This is known as the *random noisy strategy* (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). In this context, Rangel et al. (2025a), they highlight that the *random Caesar* based on the *traditional Caesar algorithm* (Rangel and Rangel 2024a; Rangel et al. 2023; Barranco and Galindo 2022; Gómez et al. 2012), it stands out due to its dynamic encryption with AI, using heuristic methods. The *traditional Caesar* cipher is based on a single static shifting value K , as follows: $C_i = S_i + K \bmod 26$ (Barranco and Galindo 2022; Gómez et al. 2012). Conversely, the *random Caesar* employs shifting values (K_i) that vary for each S_i , selected randomly. The operation: $C_i = S_i + K_i \bmod N$, it allows *random Caesar* to adapt to different N values. In the case of the *traditional Caesar* algorithm with *mod 26*, the alphabet is restricted to 26 characters for encryption and decryption purposes. The *mod N* in *random Caesar* is potentially dynamic, but it has been narrowed down to three modes in recent research (Rangel and Rangel 2024a, 2024b; Rangel et al. 2023, 2025b): *random Caesar I*, with *mod 9 & mod 255* (Rangel et al. 2025b), *random Caesar II mod 95* (Rangel and Rangel 2024a, 2024b; Rangel et al. 2023), and *random Caesar II mod 120* (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b). These schemes establish the criteria for choosing alphabets through a random methodology, using a heuristic approach to determine the optimal K_i vector. The *mod N* value, it determines the size of the encryption alphabet and the maximum value in the K_i vector. Hence, the $N=95$ value, it can obtain the ASCII characters with a range between 32 and 126 ordinal values. For example, from '(spacing)' to '~' characters. While the $N=120$ value, it can select characters with a range between 30 and 150 that they correspond to the ASCII table. Any other N value, its K_i will be selected between 0 and N ordinal value. Moreover, the *random Caesar* scheme, it consists in a second phase its procedure. It was designed for confusing the cybercriminals (Rangel and Rangel 2024a). According to (Rangel et al. 2025a), this phase corresponds to the package calculation that it can be defined as follows: $FinalPackage = C_i \& K_i \& OrdChr(C_i)$. Where, the $\&$ operator corresponds to the *concatenation* function, while the *OrdChr* procedure should put the same character of C_i in *final* of the *package* when $N=120$ mode, it is employed. Otherwise, *OrdChr* transforms the C_i to ordinal value. This method is limited to plaintext encryption as a dynamic strategy (Rangel and Rangel 2024a), while *random Caesar II mod 120* has been employed in other studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), as a means of introducing randomness. In these terms, *random noisy strategy* introduces eight alternatives (Rangel et al. 2025a), specifically *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 initiatives are based on standard encryption methods augmented by noisy injection using *random Caesar II mod 120*. In a similar vein, the *random noisy GOST* was presented in other study (Rangel et al. 2025b), and the *random noisy Camellia* (Rangel and Rangel 2025a), it was also featured. Additionally, variants of *random noisy strategies* (Rangel and Rangel 2024a), have revealed that the noisy injection schemes are known for camouflaging ciphertext (Rangel et al. 2025a). These include *reduced random Caesar* (Rangel and Rangel 2024a; Rangel et al. 2025a), and *reduced random mutation* (Rangel and Rangel 2024a). The main objective of these schemes, as outlined in other research (Rangel and Rangel 2024a), it is to camouflage and compress at least $\frac{1}{3}$ of the ciphertext. Both plaintext (Rangel and Rangel 2024a), and ciphertext (Rangel et al. 2025a), they have seen the application of the *reduced random Caesar* strategy, whereas plaintext is the sole domain of the *reduced random mutation* (Rangel and Rangel 2024a). We did not experiment with these *reduced random* and *mutation* strategies here, as they may be addressed in future work. Similarly, four *random noisy strategies* based on standard symmetric encryption algorithms, they were examined in this work, specifically

random noisy DES, random noisy 3DES, random noisy AES-256, and random noisy Blowfish. In line with this approach, a novel method known as *random noisy CAST-128* was implemented and it is here presented. Noisy injection techniques with *random noisy strategies* (Rangel et al. 2025a, 2025b; Rangel and Rangel 2024a, 2025a), they are advised for application over plaintext (Rangel and Rangel 2024a), or ciphertext (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), because they effectively indicate the performance of dynamic encryption. These schemes have not been evaluated for vulnerabilities using quantum computing (Rangel et al. 2025a, 2025c). Given that noisy injection alternatives have shown promise, this research continues the work of *random noisy strategies* (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), as mentioned above. Despite the disadvantages, we can increase performance by examining the *CAST-128* encryption algorithm's potential when applied to ciphertext, with adequate validation, a gap in existing research that could benefit organizations employing *CAST-128*. Moreover, this opens up possibilities for organizations to adopt noisy injection methods based on the *CAST-128* scheme. Therefore, we focus on cybersecurity strategies based on random noisy encryption strategies. It was carried out with the purpose of suggesting the noisy injection over of the ciphertext that it has been obtained by standard encryption algorithms. Due to that this situation may confuse to the cybercriminals (Rangel et al. 2025c). We examined only two types of situations in this research. The study began with a comparison of five standard encryption algorithms (*DES, 3DES, AES-256, Blowfish, and CAST-128*), being used as static encryption schemes, with results compared to prior studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). Second, the *random noisy CAST-128* strategy, it was introduced as a new alternative for comparative analysis with other *random noisy* alternatives (*random noisy DES, random noisy 3DES, random noisy AES-256, and random noisy Blowfish*), which they were tested for noisy injection on ciphertext. Due to that these strategies, they consist in the employment of *random Caesar II mod 120* (Rangel and Rangel 2024a), being applied on ciphertext that it has been previously obtained by *standard encryption algorithm*. Here, both goals are considered dynamic encryption alternatives to random performance. It also makes an empirical contribution to cryptography using AI, as the literature on *random noisy strategies* is sparse. Despite that recent research (Rangel and Rangel 2024a, 2024b; Rangel et al. 2023, 2024a, 2025a), they have reported that *random Caesar II* (Rangel and Rangel 2024a), it might obtain random values for the ciphertext that they can be selected out of the range of the ASCII table (Rangel et al. 2025a). In practical domains that they were evaluated with *random noisy strategies* (Rangel et al. 2025a), these problems have not been observed. These random noisy encryption strategies were previously evaluated with five-fold cross-validation (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). In this work, performance is quantified by *average* or *global accuracy* (Rangel 2002, 2022; Barandela et al. 2003; Lewis and Catlett 1994). The paper outlines the experimental results of a large-scale research project examining digital data theft scenarios. The study looked into situations where the application of at least one inadequate static encryption method compromised security (Rangel et al. 2025a). Our experiments began by exploring the replacement of static encryption schemas with *random noisy strategies* for dynamic encryption (Rangel and Rangel 2024a). Several ciphertext exemplars resulting from random noisy encryption schemes are demonstrated here. These approaches were assessed across five samples using a novel updated cross-validation (Rangel and Rangel 2024a; Rangel et al. 2024a, 2025a). The application of noisy injection on ciphertext output is suggested as it proves to be a reliable indicator of dynamic encryption efficacy. A new definition, referred to as the *random noisy CAST-128* strategy, it is here presented as an innovative dynamic data encryption method. The results are also compared against four *random noisy strategies* based on the *DES, 3DES, AES-256, and Blowfish* algorithms.

2. RELATED THEORETICAL AND EXPERIMENTAL METHODS

Replacing cybersecurity strategies with static encryption algorithms, it does not guarantee data security for organizations. In such cases, using a dynamic encryption alternative is recommended (Rangel and Rangel 2024a). This paper investigates the potential of dynamic encryption methods using *random noisy strategies* (Rangel et al. 2025a), to mitigate the problem of digital data theft. This research is experimental and exploratory in nature, introducing a novel alternative based on the *random noisy CAST-128* strategy, which it is presented here for the first time. Our research utilized a

combination of *hardware* and *software* resources. A personal computer with a 2 GHz CPU, 4 GB of RAM, and 32 GB of free disk space, it was used to conduct experiments. The encryption methods, including *DES*, *3DES*, *AES-256*, *Blowfish*, and *CAST-128*, as well as the novel variants based on *random noisy strategies* (Rangel et al. 2025a), they were implemented using Python (Python.org 2024), on Microsoft Windows 10 (Microsoft 2025). In order to compare our results with those reported in other studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), we performed some experiments on a mobile computing device with identical *hardware* specifications to the personal computer, but running Android 9 (Google 2024), and using PyDroid3 (Pydroid3 2025). The results showed no significant differences between the two platforms. We utilized *training samples* (Rangel et al. 2025b; Rangel and Rangel 2025a; Rangel 2002, 2022; Barandela et al. 2003), as our *datasets* (named here as *TS*), which consisted of 1000 randomly created exemplars. Each row in the dataset represents a *pattern* with five features or columns. Each entry includes encryption and decryption information, comprising ciphertext (C_i) in the form of a tuple (*Test1*, *Test2*), along with additional metrics such as encryption time (*TC*), decryption time (*TD*), *error* rate, and *class label*. This *pattern* can be defined as $TP = [(Test1, Test2), TC, TD, Error, Label]$, facilitating comparison with previous studies (Rangel and Rangel 2025a). As shown in *Table 1*, two ciphertext exemplars are provided for the *label* "Welcome". The *label* feature is the same plaintext (S_i), which it is simulating to be a password. It includes noisy characters: ' ' and ' ' (with ordinal values: 9619 and 65533, respectively). Encryption and decryption times, they were computed in milliseconds, while that *TC*, *TD*, and *error* features, they 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 calculating *TD*. Both sequences, along with their respective *TD*, *TC*, and *error* rates, they were stored in training sample (*TS*), using a *pattern* structure. The error rate is computed by counting the number of characters with errors. When the ciphertext generated by the encryption strategy differs from the plaintext (S_i), the error rate is computed based on the size of S_i , which it has errors. If an eight-character ciphertext has a matching eight-character plaintext but an error value of 0.5, it means that half of the characters (four from *Test1* or *Test2*), they were not decrypted accurately. Both ciphertext and plaintext are encoded as ASCII or UTF-8 character sequences, with each sequence having a maximum length of 255 characters. The *3DES* and *Blowfish* algorithms, they were exceptions, as they couldn't support sequences of 255 characters. Consequently, *Blowfish* was experimented with up to 13 characters, and *3DES* with up to 22 characters. The encryption times being evaluated mean that the comparison of algorithms, it was not entirely on equal terms. To address this, the study experimented with ciphertexts filled with random hexadecimal values to provide a more equitable assessment. This information was used to create a new format based on *cross-validation* (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b). Each encryption strategy was then executed using the updated *TS* format. To facilitate result comprehension, *Table 1* shows the *arithmetic mean* and *standard deviation* for the plaintext "Welcome", which it has been encrypted using most algorithms under the same terms. Finally, distinct *TS* were developed for each encryption algorithm experimented with, including the incorporation of random noisy strategies.

For implementing encryption methodologies, the first adopted strategy involves utilizing *standard symmetric encryption algorithms*, specifically *DES*, *3DES*, *AES-256*, *Blowfish*, and *CAST-128*. They were tested as standard encryption schemes applied to plaintext, allowing for results comparison with other research (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). These experiments were run on five *training samples* with the above-mentioned details, and each encryption algorithm was assessed separately using a *modified cross-validation* method (Rangel and Rangel 2024a; Rangel et al. 2025a). We implemented these *standard encryption algorithms* using Python (Python.org 2024), which necessitated the installation of packages like *cryptography* (Python Cryptography 2025), *pycryptodome* (PyCryptodome 2025; PyPI 2024), and *pycrypto/pycryptor*. Therefore, it needs to be imported into the source code 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, and from Crypto.Cipher import PKCS1_OAEP, ARC4, CAST ; from Crypto.Util.Padding import pad, unpad`. Let's assume that $S_i = \text{"Welcome"}$. To obtain the ciphertext based on the *DES* algorithm, the following

operation can be applied: $C_i = ((A(Key.encode()), Mode)).encrypt(plaintext) .hex(); print(C_i)$. The ciphertext obtained through the 3DES and Blowfish algorithms can be calculated as described below: $C_i = (Ri.update(plaintext) + Ri.finalize()).hex(); print(C_i)$. In a similar way, the ciphertext for the AES-256 encryption alternative can be computed as follows: $C_i = (IV.encode() + (Ri.update(plaintext) + Ri.finalize())).hex(); print(C_i)$. The ciphertext generated using the CAST-128 algorithm is presented below: $C_i = Ci.encrypt(pad(plaintext, N)).hex(); print(C_i)$. The parameters are: A (algorithm), Key (secret key), IV (initialization vector), $Mode$ (operation mode), and $plaintext$ (encoded S_i). To ensure successful decryption, the CASTPadding component must store the padding or initialization vector ($C_i.iv$). The dynamic nature of CASTPadding's ciphertext is evident in the varying results obtained from repeated encryptions. For example, three invocations of the CASTPadding.hex() function produced different outputs: '9d071f681ce59b75', '81f4aedbe675cda6', and '4b8682983930e2fd' for the $C_i.iv$ vector. Each call to the function generates a new, distinct $C_i.iv$ value, ensuring the ciphertext's dynamic behavior. This scheme supports dynamic injection into the ciphertext. The pad() function also allows for adding padding as needed. The encrypt() procedure returns a ciphertext object. The N value represents the maximum number of bytes in a character sequence. The R_i component is a partial ciphertext object without padding or one that has not yet completed processing. The Q_i component refers to the padding used. The updated() and finalize() procedures are used to complete the encryption operation. Finally, the hex() function is used to translate byte values into hexadecimal format. Given these terms, the valid parameters for DES algorithm ciphertext generation can be calculated as follows: $Key = "00000001"$; $A = DES.new$; $N = 8$; $Mode = DES.MODE_ECB$; $plaintext = Si.encode() + (b"\x00" * (N-len(Si.encode()) % N))$. For generating ciphertext using the Blowfish algorithm, the valid values are calculated as: $Key = "00000001"$; $A = algorithms.Blowfish$; $N = 16$; $Mode = modes.ECB()$; $Ri = Cipher(A(Key.encode()), Mode, default_backend()).encryptor()$; $plaintext = (Si + "" + str("".join([" " for k in range(0,int(N-len(Si.encode()))]))).encode()$. To obtain the ciphertext using 3DES, some valid values, they can be computed as follows: $Key = "00000000000000000000000000000001"$; $A = algorithms.TripleDES$; $N = 24$; $Mode = modes.ECB()$; $Ri = Cipher(A(Key.encode()), Mode, default_backend()).encryptor()$; $plaintext = (Si + "" + str("".join([" " for k in range(0,int(N-len(Si.encode()))]))).encode()$. The valid values for ciphertext generation using the AES-256 encryption version are as follows: $Key = "00000000000000000000000000000001"$; $A = algorithms.AES$; $N = 256$; $IV = "0000000000000001"$; $Mode = modes.CBC(IV.encode())$; $Ri = Cipher(A(Key.encode()), Mode, default_backend()).encryptor()$; $Q_i = padding.PKCS7(N).padder()$; $plaintext = Q_i.update(Si.encode()) + Q_i.finalize()$. The valid values for ciphertext generation with the CAST-128 algorithm can be specified as: $Key = "0000000000000001"$; $A = CAST.new$; $N = CAST.block_size$; $Mode = CAST.MODE_CBC$; $plaintext = Si.encode()$; $C_i = (A(Key.encode()), Mode)$; $CASTPadding = (C_i.iv)$. Each of the above statements must be written in the source code before calling C_i , as appropriate.

On the other hand, regarding the random noisy schemes, a second strategy employs random noisy alternatives (Rangel et al. 2025a), for dynamic data encryption, building on the idea presented by other research (Rangel and Rangel 2024a), because these schemes can effectively increase the noise in ciphertext outputs. These strategies (Rangel et al. 2025a), they 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 CAST-128 is introduced as a new proposal in this study. These five random noisy strategies, they have been implemented using Python (Python.org 2024), as mentioned above, but they were experimented with a noisy injection application that it combines the random Caesar II mod 120 being applied on the obtained ciphertext from each standard encryption algorithm separately. It was done for results comparison with other studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). The experiments were performed on five TS, as previously mentioned. Each encryption algorithm was assessed individually using the TS with a five-fold cross-validation approach (Rangel and Rangel 2024a; Rangel et al. 2025a). In our experiments, we used modified cross-validation to compute the global average and standard deviation for each encryption strategy. According to (Rangel et al. 2025a), the application of noisy injection to ciphertext is crucial, and the computation of random noisy strategies is outlined in other studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). We computed $RandomNoisy_i$ using the expression: $RandomNoisy_i = Char (Ord (StandardEncryption_i) + Ord(K_i)) \& Char (K_i) \& Char (StandardEncryption_i)$, with mod 120. By

replacing the plaintext with ciphertext, the operation was streamlined, as shown other authors (Rangel and Rangel 2024a). Moreover, it is adjusted to the *mod 120* measure because only character types, they are stored in *FinalPackage* that it is named as *RandomNoisy_i*. Therefore, several versions of *random noisy strategies*, they were presented in previous work (Rangel et al. 2025a). This research focused on four strategies for generating *StandardEncryption_i* ciphertext. We also implemented the *random noisy CAST-128* strategy, which it can be computed using the statement: $RandomNoisyCAST_i = Char (Ord (StandardCASTEncryption_i) + Ord(K_i)) \& Char (K_i) \& Char (StandardCASTEncryption_i) \bmod 120$. Where: The + operator refers to the *sum* function, while the & operator corresponds to the *concatenation* function. The K_i shifting vector, it contains random integer values. The *StandardEncryption_i* parameter, it refers to the ciphertext obtained using a *standard encryption algorithm* (e.g., *DES*, *3DES*, *Blowfish*, and *AES-256*), as described in previous research (Rangel et al. 2025a). Similarly, the *StandardCASTEncryption_i* argument refers to the ciphertext obtained through the *CAST-128* algorithm. Each strategy was applied in isolation. *RandomNoisy_i* corresponds to the *FinalPackage* resulting from a *random noisy strategy*, as mentioned in other research (Rangel et al. 2025a). In contrast, *RandomNoisyCAST_i* is the *FinalPackage* derived from applying *random noisy CAST-128*. The *Ord* function provides the ordinal value of a given character or integer, and the *Char* function generates the corresponding character in ASCII or UTF-8 based on the input value. The *random noisy* methods follow a sequential approach, where the *standard encryption algorithm* is applied to the plaintext initially, and then *random Caesar II mod 120* is applied to the resulting ciphertext (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). According to (Rangel et al., 2025a), emphasize that this approach differs from double encryption and warns against the repeated application of *standard encryption algorithms*, which could be exploited by cybercriminals using computational strategies to decrypt the data. This proposal suggests that noisy injection should not be applied uniformly to the entire ciphertext, as this could raise suspicions about the location of the noise (Rangel et al. 2025a). Applying it to only a segment of the ciphertext would present cybercriminals with the significant challenge of locating noisy characters, a problem that would remain complex and demanding even with quantum computing capabilities. Due to that these strategies, they consist in the employment of *random Caesar II mod 120* (Rangel and Rangel 2024a; Rangel et al. 2025a), being applied over ciphertext that it has been previously obtained by *standard encryption algorithm*. Hence, both goals are considered dynamic encryption alternatives to random performance (Rangel and Rangel 2024a). Utilizing *random noisy strategies* for encryption has resulted in dynamic ciphertext, which it plays a crucial role in strengthening data security in organizational contexts. Additionally, the *random Caesar II* method with *mod 120*, it is an AI-based approach, as it utilizes random and heuristic techniques to select the K_i vector of shifts (Rangel et al. 2025a). Hence, the artificial intelligence has been employed when the heuristic method, it was used for selecting K_i vector that it provides maximum values of the encryption alphabet. The heuristic method's similarity to genetic algorithm *selection* processes suggests the application of AI. This heuristic method serves as a validation technique for the selected ASCII characters (Rangel et al. 2025a), as previously discussed in other research (Rangel et al. 2025a). Thus, a genetic algorithm (GA), it is a stochastic process that involves *selection*, *crossover*, and *mutation* phases, culminating in an evaluation stage using a *wrapper* (Rangel et al. 2023, 2024a), or *fitness function* (Rangel et al. 2024a; Reddaiah 2019), for each generation of the GA (Delman 2004; Kalsi et al. 2018; Rangel et al. 2023; Rangel et al. 2024a; Reddaiah 2019; Kuncheva and Jain 1999; Skalak 1994; Clark 1994; Griindlingh and Van-Vuuren 2003; Matthews 1993). In these terms, *random Caesar* methodology (Rangel and Rangel 2024a; Rangel et al. 2025a), only *selection* procedure of GA is employed for selecting the alphabets with *mod 120* and its K_i vector of shifts. In this case, the range of K_i values are delimited between 30 and 150, for does not exceed the 255 ASCII value. Similarly, the *reduced random Caesar* strategy (Rangel and Rangel 2024a; Rangel et al. 2025a), it can use the GA *selection* procedure with *mod 120*. For *reduced random mutation* (Rangel and Rangel 2024a), the GA model's *selection* and *mutation* stages are utilized. The K_i shifting range for both schemes is delimited between 0 and 105 ordinal values to prevent exceeding the ASCII table maximum. This research did not cover *reduced random Caesar* and *reduced random mutation*, because they could be explored in future work. Regarding result evaluation, a *modified cross-validation* method (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b), it is proposed to internally bias the discrimination process,

building on previous discussions. The results of this study can guide the adoption of noisy injection as a security measure in organizations.

3. RESULTS AND DISCUSSION

All experiments were performed using *TS*, as mentioned earlier, the estimated *error*, *TC*, and *TD*, they were calculated separately for each encryption strategy. To compare results with other studies (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a), a five-fold modified *cross-validation* (Rangel and Rangel 2024a), it was applied to each *TS*. According to (Rangel and Rangel 2025a), the *traditional cross-validation* approach (Rangel 2002, 2022; Kuncheva and Jain 1999; Skalak 1994; Bruzzone and Serpico 1997; Barandela et al. 2003; Cover and Hart 1967), it involves partitioning the *TS* into five subsamples, each with a similar number of elements per class, and using one subsample as the test sample (*MC*) for model assessment. Subsequently, the remaining four subsamples (e.g., the suggested 80% of *TS*), they are joined together, forming a single subsample, to be able « to train the model », which it is subjected to evaluation, being used as if they were *new patterns*, these same contents in the *MC*. To obtain the *standard deviation* after calculating the *average* or *global accuracy*, the process is repeated five times (Lewis and Catlett 1994). This traditional procedure is not applicable to data encryption/decryption, as the evaluation of the encryption algorithm in this research does not require a training model with *TS* or evaluation with *MC*. Hence, the training and evaluation tasks, they are done before creating the ciphertext or the *FinalPackage*, according to the case. The process begins with applying the *standard encryption algorithm* to the plaintext to obtain a ciphertext. The ciphertext is then decrypted, and both vectors are saved in the *TS*. In *random noisy strategies*, the *standard encryption algorithm* is applied to the plaintext to produce a ciphertext. The heuristic method is then used to emulate partial phase training, incorporating the GA's random *selection* stage. Thereby, the K_i vector is obtained by partial training using the *random Caesar* strategy that it must be applied to the ciphertext for noisy injection in *FinalPackage*. Similarly, the encrypted sequence is decrypted, to save both vectors in the *TS*. The novel *cross-validation modification* (Rangel and Rangel 2024a; Rangel et al. 2025a), it has been updated to adopt a new approach that does not rely on *MC* for assessing *global accuracy* (Lewis and Catlett 1994). The data encryption/decryption process in this research uses a *modified cross-validation* approach (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b), which it omits *MC pattern* evaluation. The *error* is computed using four subsamples of the *TS*, and this process is repeated five times, with about 20% of the *TS* being sequentially omitted. This scheme simulates the estimated *error* in different environments by omitting a part of the *TS*, thereby it ensures a more convergent result with an optimistic bias. For example, the obtained value can be better or equal in practical applications (Rangel and Rangel 2025a). This approach allows for the assessment of the estimated *error* and other numerical features of the *TS*. If the decrypted ciphertext does not match the original plaintext, the *error* percentage is determined by calculating the proportion of coded characters that they were not decrypted correctly. In line with *traditional cross-validation*, the process is repeated five times, with sequential extraction of different subsamples. The *average* and *standard deviation* of numerical attributes or columns, including encryption times (*TC*), decryption times (*TD*), and *global error* percentage, they are computed simultaneously. Due to the observed *encryption ambiguity* during experimentation, the *cross-validation* process was conducted without class distinction. Further investigation into this matter is warranted and could be considered for future research. However, this situation has not affected the *global accuracy* of the encryption strategies here evaluated. In this study, the experimentation was confined to two classes of situations. First, the comparison of five *standard encryption algorithms* (*DES*, *3DES*, *AES-256*, *Blowfish*, and *CAST-128*), they were studied being applied to plaintext as standard encryption scheme for results comparisons with other authors (Rangel et al. 2025a, 2025b; Rangel and Rangel 2025a). In the second part, we assessed four *random noisy strategies* described by other research (Rangel et al. 2025a), being based on ciphertext. The strategies examined included *random noisy DES*, *random noisy 3DES*, *random noisy AES-256*, and *random noisy Blowfish*, with *random noisy CAST-128* being proposed as a new approach in this study. Accordingly, the five *random noisy strategies*, they were tested separately for their ability to inject noise into the ciphertext as a dynamic encryption measure. Having processed all samples for each encryption strategy

individually, the global average results were computed using the novel *cross-validation* method (Rangel and Rangel 2024a, 2025a; Rangel et al. 2025a, 2025b), as explained in the preceding sections. As shown in *Table 1*, the *standard deviation* is presented in parentheses alongside the relevant data. For the purpose of comparison with recent studies (Rangel et al. 2025a; Rangel and Rangel 2025a), the encryption and decryption times, measured in milliseconds, they are listed in columns *TC* and *TD*, respectively. The iterative procedure was stopped in this study after five repetitions of ciphertext were generated for each encryption method. As a reminder, the results shown in *Table 1* correspond to the *average* of five sequential experiments, with the *standard deviation* in parentheses, evaluated with the *updated cross-validation* technique (Rangel et al. 2023, 2024a, 2025a, 2025b, 2025c; Rangel and Rangel 2024b, 2025a). The standard encryption methods were executed with the parameters detailed below: The parameters for the *DES* algorithm consisted of a 56-bit key ('00000001'), UTF-8 encoding, *ECB* mode (Stinson and Paterson 2019), and hexadecimal output for ciphertext. The *DES* algorithm was implemented using the *pycryptodome* package in Python (PyCryptodome 2025; PyPI 2024). To obtain the ciphertext using *TripleDES (3DES)* algorithm, the *ECB* format (Stinson and Paterson 2019), using *OpenSSL* (Van-Tilborg 2005; Van and Jajodia 2011), as *default_backend()* function, and the *secret key* of 24 bits with "000000000000000000000001" value, they have been employed generating the ciphertext outputs with hexadecimal encoding. The *3DES* algorithm has been also implemented in Python using the *cryptography* package (Python Cryptography 2025). In the *Blowfish* algorithm, *ECB* format (Stinson and Paterson 2019), with *OpenSSL*-based *default_backend()* and a 16-bit *secret key* ('00000001'), they were employed for producing hexadecimal-encoded ciphertext. The implementation was carried out using Python's *Crypto.Cipher* module from *cryptography* packages (Python Cryptography 2025). For the *AES-256* algorithm, experiments used a 256-bit key ('00000000000000000000000000000001') and a 128-bit IV ('0000000000000001') with *CBC* mode (Stinson and Paterson 2019), and *OpenSSL* (Van-Tilborg 2005; Van and Jajodia 2011), *PKCS7* padding (Van-Tilborg 2005; Van and Jajodia 2011), with 128 bits was applied for hexadecimal-encoded ciphertext outputs. The scheme's implementation in Python was achieved using the *Cipher* component from the *cryptography* library (Python Cryptography 2025). For *CAST-128*, the parameters consisted of a 128-bit *secret key* ('0000000000000001'), an 8-bit block size, and *CBC* mode (*mode=2*), defined by the default values of *CAST.block_size* and *CAST.MODE_CBC* from *pycryptodome* (PyCryptodome 2025; PyPI 2024). A 16-bit *IV (Ci.iv)* was dynamically chosen for *CBC* mode (Stinson and Paterson 2019), resulting in varying outputs (e.g., '9d071f681ce59b75', '81f4aedbe675cda6', and '4b8682983930e2fd'). The algorithm used an 8-bit padding based on the *pad()* function's default settings from *pycryptodome* (PyCryptodome 2025), and the ciphertext was output in hexadecimal format.

The calculation of encryption/decryption times (in milliseconds) and estimated error, they are shown in *Table 1*. Some of them were rounded to match the results presented in other studies (Rangel et al. 2025a; Rangel and Rangel 2025a). Except for ciphertext (*Test 1* and *Test 2*) with the *random noisy strategies*, they do not match because these processes of the encryption method are random. Hence, it always generates different and dynamical results. The columns *TC* and *TD*, they are the *average* times of the estimated procedure for encryption and decryption tasks, respectively (the *standard deviation* among parentheses is shown). Two *tests* of ciphertext results for each encryption strategy, they are also shown. Two experimental *tests* with the same plaintext: "Welcome", in some cases, different results have been obtained. In contrast, the majority of the experiments yielded successful results. Some tests with the *CAST-128* and *DES* algorithms and their *random noisy schemes*, they were exceptions, as they reported errors. This could be attributed to non-ASCII characters in the plaintext, such as '█' and '□', which have ordinal values of 9619 and 65533, respectively. In real-world scenarios, controlling input data is difficult, but as shown in *Table 1*, most encryption strategies managed to obscure this problem in the ciphertext. This situation can occur in the *CAST-128* scheme if the *Ci.iv* vector is not stored correctly. According to (Rangel et al. 2025a), it is pointed out that a limitation of *random noisy* approaches is the lack of control over the maximum random ASCII value selected. Despite the success of these strategies, an ASCII value can be reinterpreted as a different encoding character, such as UTF-8. In this study, these issues were not encountered because the *mod 120* operation was used, ensuring sequences with values within the ASCII table range. Except, in particular cases with the noisy injection to the plaintext input, as mentioned above.

Average	5.2835 (3.2110)	0.5820 (0.2521)	0.4 (0.48)
---------	--------------------	--------------------	---------------

Among the traditional symmetric algorithms, the *CAST-128*-based encryption process demonstrated a significant speed advantage, being 6.18 to 6.49 times faster than the *3DES*, *AES-256*, and *Blowfish*, proposals tested. This translates to a difference of 6.12 to 6.49 milliseconds (see *Table 1*). While the *CAST-128* algorithm offers faster encryption and decryption times, it was observed in this study that it can support plaintext or ciphertext with values greater than 255 characters. Therefore, with proper validation, this limitation could not compromise the security of the scheme. In line with the *Blowfish* proposals, the *3DES* alternative has a character limit of 22 for plaintext or ciphertext and it has not reported any errors. In this work, the *CAST-128*, and *DES* algorithms, they were limited to handling plaintext or ciphertext of up to 255 characters. Due to the observed error rate of 1.0% during data decryption, they are not recommended to use the *CAST-128*, and *DES* alternatives. *Table 1* in this research displays static ciphertext results for *standard encryption algorithms*, including *DES*, *3DES*, *AES-256*, and *Blowfish*. It's crucial to understand that the static nature of these ciphertexts does not necessarily imply a lack of security in all cases. However, the *CAST-128* scheme is characterized by its *dynamic ciphertext* outputs. Furthermore, *random Caesar* on plaintext provided the optimal balance. The application of *random Caesar II* with *mod 120* resulted in faster encryption times than the other evaluated methods. 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. The combination of *CAST-128* with *random Caesar II mod 120*, named here as the *random noisy CAST-128* strategy, it has exhibited faster performance on ciphertext compared to the rest *random noisy strategies* assessed. Similarly, the *DES* schemes but it does not include its *random noisy strategy* yield the most balanced results. The novel *random noisy CAST-128* alternative outperformed the rest *random noisy strategies*, with a speed of 1.05 to 5.47 times. This resulted in a difference of 0.08 to 6.91 milliseconds. The comparison between standard *CAST-128* and its *random noisy CAST-128* version, it shows that the difference in encryption speed is not considerable. Specifically, *traditional CAST-128* differ the *random noisy strategy* by a factor of 1.30, with a distance of only 0.36 milliseconds (see *Table 1*). In this work, both *CAST-128* alternatives have shown that they support a maximum length of plaintext with 255 characters, as mentioned above. Besides, in experiments, this situation has reported errors. Regardless, it is believed that this measure alone substantially improves the trustworthiness of the encryption strategy's performance. Thus, *random noisy* schemes prove to be a valuable resource for experimentation, although further analysis of other factors is still needed due to the limited depth of the experiments, which they were conducted with plaintext inputs up to 255 characters. However, the *3DES* and *Blowfish* schemes, they are the exceptions, for the reasons outlined above. Each encryption strategy was evaluated using a separate training sample designed specifically for it. *Random Caesar* schemes have been shown to increase data security in organizations (Rangel and Rangel 2024a, 2024b; Rangel et al. 2023, 2024a, 2025a), and the results of *random noisy strategies* yield similar positive outcomes. *Table 1* illustrates this phenomenon in the computed *global average*. *Random noisy* alternatives, in particular, generate ciphertexts of slightly greater length. Other aspect that it was observed during the experiments, it refers that when ciphertext is produced by *DES* scheme, it can be faster than the *CAST-128*, *3DES*, *AES-256*, and *Blowfish* proposals here evaluated. However, the obtained ciphertext by *DES* algorithm, it can produce vulnerability to the data security in the organisations and it might be very good for the cybercriminal decryption strategy. Due to that it was observed in experiments that encryption/decryption process, it has also reported an error rate of 1.0% on average. However, when *random noisy strategies* were applied to ciphertext of *standard encryption algorithms*, they have reported dynamic encryption results in all the cases. Although that the times for obtaining the ciphertext of *FinalPackage* using *random noisy strategies*, they are also larger than the obtained times with their *standard encryption algorithms*, as appropriate (see *Test1* and *Test2* in *Table 1*). *CAST-128*

proposals exhibit faster execution times than *DES*, *3DES*, *AES-256*, and *Blowfish* strategies. Despite this, the *random noisy* schemes always generate dynamic ciphertext outputs. Cybercriminals would encounter significant obstacles in decrypting data, as they would need to determine each random K_i shifting value in advance, which it has been previously hidden. Furthermore, the novel *partial noisy injection* scheme presented in other research (Rangel et al. 2025a), referred to as the *PartialNoisy* proposal, significantly complicates the decryption process when employed. Hence, these tasks of data discovering might be very difficult, even for quantum computing. We did not perform experiments with *PartialNoisy*, leaving them as a potential area for future works. Through the repeated application of these *random noisy strategies*, we can obtain more dynamic and superior encryption results than those achieved with traditional standard algorithms. This approach not only increases the security of ciphertext and plaintext but also offers a potential defense against future quantum computing attacks (Rangel et al. 2025c; Baklaga 2024; Bavdekar et al. 2023; Iavich et al. 2024), enhancing digital data security for organizations. Furthermore, these *random noisy* alternatives, they have not been tested against any type of cyberattack to assess potential vulnerabilities. According to other research (Rangel and Rangel 2024a; Rangel et al. 2025a), it is recommended the downsized ciphertext with *reduced random* or *reduced mutation* methodologies (Rangel and Rangel 2024a), which they can be considered very good indicators. Given that these schemes can help us obtain too short the ciphertext and fast encryption process. Besides, they do not put in risk data security of the information because the partial ciphertext is camouflaged its K_i shifting before being stored into *FinalPackage*. By injecting a substantial proportion of noise into the ciphertext, these alternatives can reduce the risk of digital data theft. In other words, it is better by considering the proposals based on *reduced random* and *reduced mutation* schemes. Regardless, in this work, the *reduced* and *mutation* alternatives, they have not been employed because they can be considered a future works. The research overcame its challenges and successfully achieved its goals and hypotheses. This novel proposal, based on noisy injection, leverages *random noisy CAST-128* to produce dynamic encryption ciphertext outputs. As a result, the same input can produce different outcomes, which can create uncertainty and confusion for potential cyber threats. This information can be corroborated in *Table 1*. The comparison of results between *standard encryption algorithms* and *random noisy strategies* suggests that noisy injection can be a secure option for organizations. This encompasses the use of the novel *random noisy CAST-128* strategy, specifically in settings where *traditional CAST-128* has been adopted. The *random noisy* scheme is suggested for increasing digital data security in cryptosystems of this nature. Despite the potential downsides of the *random noisy CAST-128* encryption strategy. Organizations might consider this novel alternative a safe measure due to its ability to guarantee improved digital data security through noisy injection, providing a wide range of opportunities for its use. In addition, a modification of the dynamic encryption methodology here presented might be of interest. This technique involves applying *reduced noisy strategies* (Rangel et al. 2025a), and *reduced random mutation* (Rangel and Rangel 2024a), to ciphertext camouflage, yielding a 33% reduction in ciphertext size in *FinalPackage* relative to random noisy methods. Another strategy that could be examined is the application of different methods for noisy injection. Especially, those that merge the simultaneous *random noisy* methodology with artificial intelligence utilizing the *nearest neighbor rule* (Rangel et al. 2024a, 2025c; Rangel and Rangel, 2024b; Rangel 2002, 2022; Barandela et al. 2003; Cover and Hart 1967), and pseudo-hexadecimal encoding (Rangel et al. 2023, 2024a, 2025c), as mentioned above. Clearly, this suggests a multitude of potential directions for future work.

4. CONCLUSIONS AND FUTURE WORKS

One of the key factors influencing digital data safety in organizations is the regular updating of cybersecurity strategies, such as encryption methods. Yet, it does not provide a foolproof guarantee for digital data security. According to (Rangel and Rangel 2024a; Rangel et al. 2025c), they mention that there exists several studies that they have been involved with the problems produced by the presence of the vulnerable encryption measures. If a strategy is widely recognized, it can lose its effectiveness. Hence, in previous research (Rangel and Rangel 2024a, 2024b, 2025a; Rangel et al. 2023, 2024a, 2025a, 2025b, 2025c), some dynamical encryption alternatives, they have been recommended for handling to the theft of digital data problem. This paper introduces a new

modification of existing methodologies, combining *standard encryption algorithms* with a *random Caesar strategy*, for practical applications in organizational settings. A novel dynamical encryption strategy, it was here designated as *random noisy CAST-128*, is put forth in this study. Furthermore, the evaluation included a comparison of five dynamic encryption alternatives that utilize *random noisy strategies*. The strategies incorporate AI-based noisy injection into ciphertext, utilizing random and heuristic methods as previously described. Consequently, it can be effectively utilized in real-world organizational applications. Given its ability to address the weaknesses of *standard encryption methods*, the noisy injection methodology can significantly enhance its usefulness and applicability. Experimental results with the dynamic encryption random noisy alternatives, they have revealed that can cope with the cyberattacks and data security problems with high levels of trust. In every instance, the *random noisy strategies* outperform *standard encryption algorithms* in terms of dynamic and generalized results (see *Table 1*). Despite that the *standard CAST-128* has reported dynamic ciphertext, it has shown errors because the *Ci.iv* could not be handled adequately. This topic requires further study, which we intend to undertake. One approach we'll be looking into is the use of techniques to minimize the size of ciphertext obtained through *random noisy strategies*. It is possible to employ *reduced random* scheme (Rangel et al. 2025a), or *reduced mutation* strategy (Rangel and Rangel 2024a), given their ability to camouflage ciphertext operations. A further possibility is to examine different approaches for achieving noisy injection. In particular, the methods that synergize random noisy approaches with *nearest neighbor rule*-based AI and *pseudo-hexadecimal* encoding, as mentioned earlier. This development brings forth a broad spectrum of possibilities that we will investigate in future works.

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. Delman, B. 2004. "Genetic Algorithms in Cryptography". M.S. thesis [Thesis for the Degree of Master of Science in Computer Engineering], Department of Computer Engineering, Rochester Institute of Technology, RIT Scholar Works, Rochester, New York. (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=scholar, 23/03/2024).
2. 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 and Signal Processing*, in *Journal of Medical Systems*, [e-ISSN: 1573-689X], vol. 42, no. 17, december. DOI: <https://doi.org/10.1007/s10916-017-0851-z>.
3. Mendoza, J.C. 2008. "Demostración De Cifrado Simetrico Y Asimétrico", [Universidad Politécnica Salesian. Cuenca, Ecuador. ISSN: 1390-650X], *Ingenius, Revista de Ciencia y Tecnología*, núm. 3, pp. 46-53. (<http://www.redalyc.org/articulo.oa?id=505554806007, 09/11/2024>).
4. Rangel, E., and Rangel, K.U. 2024. "Novel Random Encryption Methods Based On Mutation Strategies Of Artificial Intelligence", *SPCSJ: Scientific and Practical Cyber Security Journal* [Published by Scientific Cyber Security Association in Tbilisi, Georgia], ISSN: 2587-4667, vol. 8, no. 3, pp. 84-91, september issue. (<https://journal.scsa.ge/issue/september-2024/, 05/11/2024>).
5. Rangel, E., Rangel, K.U., 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. (<https://journal.scsa.ge/papers/dynamic-encryption-methods-based-on-noisy-injection-and-camouflaging-ciphertext-strategies-with-artificial-intelligence/, 23/04/2025>).
6. Rangel, E., Rangel, K.U., and González, L. 2025. "Inyección De Ruido Para Encriptado De Datos Dinámico Con Inteligencia Artificial. Caso De Estudio: Algoritmo GOST R 34.12-2015", *Revista Electrónica de Divulgación de la Investigación del SABES* [Revista de la universidad del SABES, Editada por Sistema Avanzado de Bachillerato y Educación Superior en el Estado de Guanajuato. México, ISSN: 2007-3542], vol. 29, edición junio, in press. (<https://sabes.edu.mx/revista-electronica/, 11/07/2025>).
7. Rangel, E., and Rangel, K.U. 2025. "Mejorando la seguridad del algoritmo Camellia, mediante la inyección de ruido sobre textos cifrados utilizando procesos basados en inteligencia artificial", *INTELETICA, Revista de Inteligencia*

- Artificial, Ética y Sociedad [Edited & Published by IBERAMIA, Iberoamerican Society of Artificial Intelligence – Sociedad Iberoamericana de Inteligencia Artificial in Valencia, España, ISSN: 3020-7444], vol. 2, no. 3, June issue, Manuscript in review, unpublished. (<https://inteletica.iberamia.org/index.php/journal/workflow/access/45>, 05/07/2025).
8. Rangel, E., Rangel, K.U., González, L., Ortiz, A., and Rodríguez, C.A. 2025. "Four Dynamic Encryption Alternatives With Artificial Intelligence Based On Pseudo-Hexadecimal Noisy Injection Schema For Handling The Theft Of Digital Data Problem", SPCSJ, Scientific and Practical Cyber Security Journal [Edited by SCSA - SPCSJ - BOAI, Published by Scientific Cyber Security Association in Tbilisi, Georgia, e-ISSN: 2587-4667], vol. 9, no. 3, pp. 59-77, June. <https://journal.scsa.ge/issue/june-2025/>, (<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/>, 27/07/2025).
 9. Rangel, E., Rangel, K.U., Medrano, J., Bernal, C.A., 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 Ciudad Altamirano, Sexto (VI) Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas, Cd. Altamirano, Estado De Guerrero, México. Noviembre. (<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).
 10. Rangel, E., Rangel, K.U., 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, Editada por Sistema Avanzado de Bachillerato y Educación Superior en el Estado de Guanajuato. México, ISSN: 2007-3542], vol. 28, edición diciembre. (<https://sabes.edu.mx/revista-electronica/27/#>, 17/12/2024).
 11. Rangel, E., and Rangel, K.U. 2024. "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 [Editado por IBERAMIA, Iberoamerican Society of Artificial Intelligence – Sociedad Iberoamericana de Inteligencia Artificial, España, ISSN: 3020-7444], vol. 1, num. 2, Dec., pp. 1–15, (<https://inteletica.iberamia.org/index.php/journal/article/view/16>, 23/03/2025).
 12. Álvarez, D. 2019. "Algunos Aspectos Jurídicos Del Cifrado De Comunicaciones", Derecho PUCP [Pontificia Universidad Católica del Perú], núm. 83, 2019, pp. 241-264. DOI: <https://doi.org/10.18800/derechopucp.201902.008>, (<http://www.redalyc.org/articulo.oa?id=533662765008>, 09/11/2024).
 13. Barranco, F., and Galindo, C. 2022. "Criptografía básica y algunas aplicaciones". Universidad Jaume I, Departamento de Matemáticas, Castellón, España, octubre. URI: <http://hdl.handle.net/10234/201359>, (<https://repositori.uji.es/items/35da2f29-ee4a-4dbc-a82f-c450a81cf9be>, 13/04/2025).
 14. Gómez, S., Arias, J.D., and Agudelo, D. 2012. "Cripto-Análisis Sobre Métodos Clásicos De Cifrado". Scientia Et Technica, [Universidad Tecnológica de Pereira Pereira, Colombia, ISSN: 0122-1701, Año: XVII], vol. 2, no. 50, pp. 97-102, abril, DOI: <https://doi.org/10.22517/23447214.6681>, (<https://revistas.utp.edu.co/index.php/revistaciencia/article/view/6681>, 16/04/2025).
 15. Javidi, B., and Horner, J.L. 1994. "Optical Pattern Recognition for Validation and Security Verification", Optical Engineering [ISSN: 0091-3286], vol. 33, issue: 6, pp. 1752-1756, June, DOI: <https://doi.org/10.1117/12.170736>, (<https://www.spiedigitallibrary.org/journals/optical-engineering/volume-33/issue-6/0000/Optical-pattern-recognition-for-validation-and-security-verification/10.1117/12.170736.short>, 14/04/2025).
 16. Reddaiah, B. 2019. "A Study on Genetic Algorithms for Cryptography", International Journal of Computer Applications, [Department of Computer Applications, Yogi Vemana University Kadapa, A.P, India, ISSN: 0975-8887], vol. 177, no. 28, pp. 1-4, december. DOI: <http://dx.doi.org/10.5120/ijca2019919509>, (https://www.researchgate.net/publication/338012809_A_Study_on_Genetic_Algorithms_for_Cryptography, 23/03/2024).
 17. Sebas, C. 2023. "¿Qué son los Algoritmos Genéticos en las Inteligencias Artificiales?", Manuales y Tutoriales de Informatica., (<https://aprendeinformaticas.com/ia/>, 23/03/2024).
 18. Paul, S., Dasgupta, P., Naskar, P.K., 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, June issue. DOI: 10.18280/rces.040206, (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=2ahUKewjVIYfordOMAxVVLUQHIXknBWsQFnoECB4QAQ&usg=AOvVaw3yvem4PsuKbMM9009owuAJ, 12/04/2015).
 19. Opplinger, R. 2005. "Contemporary cryptography", 1ra. ed., Artech House Computer Security Library, [Boston/London, ISBN: 1-58053-642-5, 978-8189265038], 510p. (<https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://theswissbay.ch/pdf/Gentoomen%2520Library/Cryptography/Contemporary%2520Cryptography%2520-%2520Rolf%2520Opplinger.pdf&ved=2ahUKewjT1tm1xNWMxUAmO4BHWYLNaoQFnoECBoQAQ&usg=AOvVaw3GvxLi6QzJhvEXqcl9efPz>, 13/04/2025).

20. Stinson, D.R. and Paterson, M.B. 2019. "Cryptography: Theory and Practice", (4ta ed.). Chapman and Hall Book/CRC Press [Taylor & Francis Group, ISBN: 978-1-1381-9701-5]. (https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.ic.unicamp.br/~rdahab/cursos/mo421-mc889/Welcome_files/Stinson-Paterson_CryptographyTheoryAndPractice-CRC%2520Press%2520%25282019%2529.pdf&ved=2ahUKEwi8_euUtdWMAxWzh-4BHSgGAaQQFnoECCUQAQ&usg=AOvVaw1DCBGkDBaZMeHuyR45xZZC, 13/04/2025).
21. Van-Tilborg, H.C.A. 2005. "Encyclopedia Of Cryptography And Security", TUE Research portal. Springer, pp. 114-115, 201-202. DOI: <https://doi.org/10.1007/0-387-23483-7>, (09/11/2024).
22. Baklaga, L. 2024. "Leading The Way In Quantum-Resistant Cryptography For Everyday Safety". SPCSJ, Scientific and Practical Cyber Security Journal [Scientific Cyber Security Association (SCSA).Tbilisi, Georgia, ISSN: 2587-4667], vol. 8, no. 3, pp 65-73, issue. (<https://journal.scsa.ge/papers/leading-the-way-in-quantum-resistant-cryptography-for-everyday-safety/>, 23/03/2025).
23. 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).
24. Dang, Q.H., and Le, H.Q. 2022. "Improved cryptanalysis of the RSA algorithm using side-channel attacks", Journal of Information Security and Applications, vol. 65, no. 103313.
25. Luciano, D., and Prichett, G. 1987. "Cryptography: From Caesar Ciphers To Public-key Cryptosystems". The College Mathematics Journal, vol. 18, pp. 2-17. (<http://www.jstor.org/stable/2686311>, 06/06/2025).
26. Rahman, M.S., and Hossain, M.S. 2021. "A Secure Private Key Cryptography Scheme Using RSA and AES". Journal of Cybersecurity, 1(1), pp. 1-9.
27. 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. (<https://repository.udistrital.edu.co/handle/11349/28192>, 06/05/2024).
28. Baker, M., and Schiller, J. 2015. "ECIES: Elliptic Curve Integrated Encryption Scheme", In Cryptography and Network Security, Springer, pp. 245-263. (<https://medium.com/asecuritysite-when-bob-met-alice/elliptic-curve-integrated-encryption-scheme-ecies-encrypting-using-elliptic-curves-dc8d0b87eaa>, 23/03/2025).
29. Hankerson, D., Hernandez, J.L., and Menezes, A.J. 2004. "Software implementation of elliptic curve cryptography over binary fields", Springer-Verlag Berlin Heidelberg 2000, [K. Ko,c and C. Paar (Eds.) – CHES 2000, LNCS 1965], pp. 1–24. (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).
30. Montgomery, P.L. 1987. "Speeding up the Pollard rho method", Mathematics of Computation, vol. 48, no. 177, pp. 453-456.
31. NIST. 2013. "Special Publication 800-56A Revision 2: Recommended methods for key establishment using public key cryptography", NIST Special Publication 800-56A. (<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=2ahUKEwio1eCz3KGMaXWgJ0QIHc4BXQQFnoECBsQAQ&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/ar2/final&ved=2ahUKEwio1eCz3KGMaXWgJ0QIHc4BXQQFnoECBwQAQ&usg=AOvVaw183JO5CLTxYuENINrMwqg>, 23/03/2025).
32. Dhany, H.W., Izhari, F., Fahmi, H., Tulus, M., and Sutarman, M. 2018. "Encryption and Decryption using Password Based Encryption, MD5, and DES", Published by Atlantis Press, [ISSN: 2352-5398. Open Access: CC BY-NC], license: <http://creativecommons.org/licenses/by-nc/4.0/>.
33. 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), pp. 555-564.
34. Van, H.C., and Jajodia, S. 2011. "Encyclopedia Of Cryptography And Security". Springer Science & Business Media, [ISBN: 978-14419-5907-2], 1416p.
35. Schneier, B. 1994. "Description of a new variable-length key, 64-bit block cipher (Blowfish)", In Fast Software Encryption.
36. Schneier, B. 2000. "Secrets and lies: Digital security in a networked world". Wiley.
37. AL-Maqtari, E.A., and AL-Maqtari, E.A. 2024. "Performance Evaluation for AES, Blowfish, DES, and 3DES Cryptography Algorithms", PUIRP: Partners Universal Innovative Research Publication (ISSN: 3048-586X, Tamilnadu, India), vol. 2, no. 5, pp. 86-95, october, doi: <https://doi.org/10.5281/zenodo.13974870>. Available at: (<https://www.puirp.com/index.php/research/article/view/81>, 27/04/2025).
38. Muhammed, R.K., Aziz, R.R., Hassan, A.A., Aladdin, A.M., Saydah, S.J., Rashid, T.A., and Hassan, B.A. 2024. "Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption", Kurdistan Journal of Applied Research (Published by Sulaimani Polytechnic University, ISSN: 2411-7684), vol. 9, no. 1, pp. 52–65, may issue, doi: <https://doi.org/10.24017/science.2024.1.5>, URL: <https://doi.org/10.48550/arXiv.2407.16274>. Available at: (<https://arxiv.org/abs/2407.16274>, 01/06/2025).

39. Wang, S., Cui, T., Wang, M. 2016. "Improved Differential Cryptanalysis of CAST-128 and CAST-256", In Information Security and Cryptology. Inscrypt. Lecture Notes in Computer Science [Chen, K., Lin, D., Yung, M. (eds), Springer, Cham, ISBN: 978-3-319-54705-3], vol 10143, march, 2017, doi: https://doi.org/10.1007/978-3-319-54705-3_2. (https://link.springer.com/chapter/10.1007/978-3-319-54705-3_2, 01/07/2025).
40. Saini, A., Tsokanos, A., and Kirner, R. 2023. "CryptoQNRG: a new framework for evaluation of cryptographic strength in quantum and pseudorandom number generation for key-scheduling algorithms", The Journal of Supercomputing (e-ISSN: 1573-0484), vol. 79, pp. 12219–12237, july, doi: <https://doi.org/10.1007/s11227-023-05115-4>. Available at: (<https://link.springer.com/article/10.1007/s11227-023-05115-4>, 14/04/2025).
41. 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).
42. Iavich, M., Kuchukhidze, T., and Gagnidze, A. 2024. "Post-quantum Digital Signature Using Verkle Trees And Lattices", SPCSJ, Scientific and Practical Cyber Security Journal [SCSA, Scientific Cyber Security Association, Tbilisi, Georgia, ISSN: 2587-4667], vol. 8, no. 3, pp. 35-52. (<https://journal.scsa.ge/issues-archive/>, 23/03/2025).
43. Fuegner, P. 2024. "Are RSA and AES Both at Risk From the Quantum Threat?", QuSecure, Inc. (<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>).
44. Sengupta, S., and Ghosh, S. 2023. "Quantum Computing Encryption Threats: Why RSA and AES Are at Risk", Journal of Cryptographic Research.
45. 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, pp. 6-12.
46. Rangel, E. 2002. "Vecinos Envolventes para Variantes de la Regla del Vecino más Cercano", [MSc.Thesis], Instituto Tecnológico de Toluca, Metepec, México.
47. 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", In Quinto (V) Congreso Nacional De Investigación En Ciencia E Innovación De Tecnologías Productivas. Tecnológico Nacional De México, campus: Instituto Tecnológico de Cd. Altamirano, Estado De Guerrero, México, noviembre. (<http://erangel.coolpage.biz/pappers/edgarrangel2022.pdf>, 06/06/2025).
48. Iyengar, S.S., Kannan, R., and Ganapathi, S. 2021. "Inteligencia artificial y criptografía: Tendencias y desafíos", IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 2, pp. 833-844.
49. Liu, X., and Wang, X. 2022. "Quantum cryptanalysis of lattice-based cryptographic protocols", Physical Review X, vol. 12, no. 2 (021004).
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, vol. 67, no. 2, pp. 257-275.
51. Mitchell, T.M. 2020. "Machine learning", (2^a ed.), McGraw-Hill.
52. Ross-Quinlan, J. 1993. "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, CA.
53. Russell, S.J., and Norvig, P. 2020. "Inteligencia artificial: Un enfoque moderno", (4^a ed.), Pearson.
54. Hartmann, A.K. 2020. "Heuristic search in graphs", Journal of Artificial Intelligence Research, vol. 68, pp. 1-33.
55. Sánchez, J.S., Pla, F., and Ferri, F.J. 1997. "Prototype selection for the nearest neighbor rule through proximity graphs", Pattern Recognition Letters 18, pp. 507-513.
56. Kuncheva, L.I., and Jain, L.C. 1999. "Nearest Neighbor Classifier: Simultaneous editing and feature selection", Pattern Recognition Letters, vol. 20, pp. 1149-1156.
57. Murphy, K.P. 2022. "Probabilistic machine learning: An introduction", MIT Press.
58. Reddaiah, B. 2016. "A Study on Pairing Functions for Cryptography", IJCA (0975-8887), vol. 149, no. 10, pp. 4-7, september.
59. Skalak, D.B. 1994. "Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms", In ML94, Proceedings of the Eleventh International Conference on Machine Learning, [Morgan Kaufmann], pp. 293-301.
60. 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.
61. Griindlingh, W., and Van-Vuuren, J.H. 2003. "Using Genetic Algorithms to Break a Simple Cryptographic Cipher", (submitted 2002), Retrieved March 31, unpublished. (http://dip.sun.ac.za/~vuuren/abstracts/abstr_genetic.htm).
62. Matthews, R.A.J. 1993. "The use of genetic algorithms in cryptanalysis", Cryptologia, vol. 17, no. 4, pp. 187-201.
63. 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]. DOI: [https://doi.org/10.1016/S0167-8655\(97\)00109-8](https://doi.org/10.1016/S0167-8655(97)00109-8). (<https://www.sciencedirect.com/science/article/abs/pii/S0167865597001098>, 01/06/2025).
64. Goodfellow, I., Bengio, Y., and Courville, A. 2021. "Deep learning", MIT Press.
65. Barandela, R., Sánchez, J.S., García, V., and Rangel, E. 2003. "Strategies for Learning in Class Imbalance Problems", Pattern Recognition [Rapid and Brief Communication, Pergamon, ISBN: PII: S0031-3203(02)00257-1.0031-3203(02)], vol. 36, no. 3, pp. 849-851. DOI: [https://doi.org/10.1016/S0031-3203\(02\)00257-1](https://doi.org/10.1016/S0031-3203(02)00257-1), (01/06/2025).

66. Lewis, D., and Catlett, J. 1994. "Heterogeneous Uncertainty Sampling for Supervised Learning", Proceedings of the 11th International Conference on Machine Learning, ICML'94, [New Brunswick, New Jersey, Morgan Kaufmann], pp. 148-156.
67. Cover, T.M., and Hart, P.E. 1967. "Nearest Neighbor Pattern Classification", IEEE Transactions on Information Theory, [e-ISSN: 1557-9654], vol. IT-13, january, pp. 21-27. DOI: 10.1109/TIT.1967.1053964. (<https://ieeexplore.ieee.org/abstract/document/1053964>, 23/03/2024).
68. Rangel, E., and Rangel, K.U. (in review, 2024). "Novel Pseudo-Hexadecimal Encryption Strategies For Camouflaging Ciphertext Based On Nearest Neighbor With Artificial Intelligence". Send to IJCOPI, International Journal of Combinatorial Optimization Problems and Informatics, ISSN: 2007-1558, manuscript in review since 2024. <https://ijcopi.org/ojs/authorDashboard/submission/529>, unpublished.
69. Microsoft. 2025. "Descarga de software", Microsoft, 2025. (<https://www.microsoft.com/es-mx/software-download>, 01/06/2015).
70. Python.org. 2024. "The Python Network", Python.org, 2024. (<https://www.python.org/downloads/>, 18/11/2024).
71. Google, Android 12. 2024. "Sistema operativo para dispositivos móviles", Google. (https://www.android.com/intl/es_es/android-12/, 01/06/2025).
72. Pydroid3 versión 7.4_arm64. 2025. "IDE for Python 3. Lenguaje de programación y compilador", Google Play Store. (<https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&pli=1>, 01/06/2025).
73. Python, Cryptography, "Cryptography 45.0.4", Python Software Foundation, 2025. (<https://pypi.org/project/cryptography/>, 01/06/2025).
74. PyCryptodome. 2025. "Crypto.Cipher package. Introduction", Readthedocs.io. (<https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html>, 30/03/2025).
75. PyPI. 2024. "Pycryptodome 3.21.0", Python Software Foundation. (<https://pypi.org/project/pycryptodome/>, 13/12/2024).