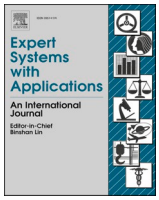
Expert Systems With Applications 254 (2024) 124470

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Securecipher: An instantaneous synchronization stream encryption system for insider threat data leakage protection

Isabel Herrera Montano [a,*], Juan Ramos Diaz [b], José Javier García Aranda [b], Sergio Molina-Cardín [b], Juan José Guerrero López [b], Isabel de la Torre Díez [a]

[a] Department of Signal Theory and Communications and Telematics Engineering University of Valladolid, Paseo de Belén, 15, 47011 Valladolid, Spain
[b] Department of Innovation, Nokia, Maria Tubau Street, 9, 28050 Madrid, Spain

ARTICLE INFO

Keywords:
Encryption algorithm
Decryption algorithm
Secure file system
Insider threat
Information security

ABSTRACT

The paper addresses the persistent challenge of insider threat in cybersecurity. Despite advancements in detection and prevention technologies and approaches, the complexity of digital environments and the ingenuity of insiders remain a problem. We propose an encryption algorithm called Securecipher, specifically designed to protect file systems from insider threats. The requirements that an algorithm must meet in this context are outlined, along with a method for its development. A context-based key generation mechanism is introduced, eliminating the need for key storage. A file marking mechanism is proposed that enables protection of the encryption algorithm against specific insider threat attacks. The proposed encryption algorithm meets the requirements posed by insider threats and has successfully passed 87 % of the NIST tests, equivalent to 13 out of 15 tests passed. Compared to conventional algorithms, the proposed encryption algorithm is more efficient in the context of insider threats, allowing access to distant locations instantaneously. In the specific case of the comparison with the RC4 algorithm, it showed a 0.25 s higher speed when accessing the last position of a 128-bytes file. Furthermore, a significant increase in the vocabulary of the encrypted text with Securecipher compared to the original text is observed, approximately 42 times more.

1. Introduction

In an increasingly interconnected world heavily reliant on electronic data management, information security becomes an unavoidable priority. The dynamic nature of cyber threats and the myriad of channels through which data can leak require advanced and proactive solutions to mitigate external and internal threats (Bandari, 2021). In this context, external threats refer to cyber-attacks caused by agents external to the organization, such as phishing, malware, hacking, among others (Herrera Montano et al., 2022). In contrast, internal threats arise from within the organization. This type of threat includes incidents caused by people with authorized access to information, either consciously or unconsciously (Pal,Chattopadhyay,and Swarnkar, 2023; Renaud et al., 2024). In the past, Data Loss Prevention (DLP) systems have emerged as essential tools to protect sensitive information. These solutions can address internal threats, negligence or malicious acts by employees, as well as external threats that seek to exploit vulnerabilities in network security (Ahmad,Mehfuz,and Beg, 2022; Herrera Montano et al., 2022).

DLP systems are based on various protection methods, from automated data classification and constant monitoring of user activities to the enforcement of access policies and information encryption. Since the early 1980 s, Secure File Systems (SFS) received considerable attention as DLP tools (Gudes, 1980). A SFS refers to a data storage environment that implements robust measures and controls to protect the integrity, confidentiality, and availability of stored information. This requires the integration of other security techniques, where the importance of encryption algorithms and secure information access techniques is highlighted (Montano et al., 2022). The key to these systems lies in their ability to adapt to heterogeneous business environments to ensure data protection (Ahmad,Mehfuz,and Beg, 2023; Faheem et al., 2017; Ghouse, Nene, and VembuSelvi. 2019).

The effective adoption of the encryption algorithm in an SFS strengthens the defenses against external and internal attacks (Faheem et al., 2017; Herrera Montano et al., 2022). During the last decades, the

* Corresponding author.
E-mail addresses: isabel.herrera.montano@uva.es (I. Herrera Montano), juan.ramos_diaz.ext@nokia.com (J. Ramos Diaz), jose_javier.garcia_aranda@nokia.com (J.J. García Aranda), sergio.molina_cardin.ext@nokia.com (S. Molina-Cardín), juan.guerrero_lopez.ext@nokia.com (J.J. Guerrero López), isator@tel.uva.es (I. de la Torre Díez).

https://doi.org/10.1016/j.eswa.2024.124470
Received 10 March 2024; Received in revised form 21 May 2024; Accepted 7 June 2024
Available online 12 June 2024

efficiency against external threat of conventional cryptographic algorithms (Djordjevic, 2019; Yegireddi and Kiran Kumar, 2016) such as: Data Encryption Standard (DES) (Nie and Zhang, 2009; Smid and Branstad, 1988; Tihanyi, 2022; Zhao, 2023), Advanced Encryption Standard (AES) (Ahmad et al., 2022,2023; Dworkin, 2023), Rivest–Shamir–Adleman (RSA) (Burnett and Paine, 2001; Thakare et al., 2024), Rivest Cipher 4 (RC4) (Paul, 2018; Stallings, 2005), and other conventional algorithms (Abujoodeh,Tamimi,and Tahboub, 2023; Bernstein, 2024; Faheem et al., 2017) has been demonstrated. The security of these algorithms is reduced in the face of the insider threat, because the insider has the information in the clear and can generate a set of documents to obtain key information (Huang and Li, 2017).

A study conducted until July 2022 (Herrera Montano et al., 2022), revealed that 40 % of the studies found in the literature concerning DLP systems, address information encryption techniques as a fundamental measure against data leakage. Similar studies such as (Ahmad et al., 2023) and (Ahmad et al., 2022), propose cryptographic approaches, based on the AES algorithm, for greater security of data transfer and storage in the cloud. In (Dhanuja et al., 2020) proposed E-REA, an extension of the Reverse Encryption Algorithm (REA) algorithm proposed in 2013 by the authors of (Bhagat, Satpute, and Palekar, 2013). The algorithm proposed in (Dhanuja et al., 2020), seeks to improve the security of data in transit.

Other recent similar studies, focus on the development of encryption algorithms for SFS, such is the case of (Bhondve, 2023), which proposes a system that employs the combination of AES and Blowfish algorithms for the development of a secure file storage system in cloud environments. The authors of (Ms. S. Suma et al., 2023) propose hybrid cryptography for a secure storage system, the proposed encryption mechanism uses the combination of multiple conventional encryption algorithms. The study described in (Adeniyi et al., 2023) focuses on modifying the AES algorithm to achieve better performance in protecting patient medical information stored in the cloud. However, an encryption algorithm for DLP systems focused on the special needs of the insider threat has not been proposed so far.

The main objective of this paper is to propose Securecipher, a new encryption algorithm for SFS, valid to mitigate the insider threat. In this research work, we study the main requirements of a valid encryption algorithm to mitigate the insider threat. A mechanism based on boolean functions is proposed for the development of this type of encryption algorithms. Additionally, an encryption algorithm, with context-based key generation, is presented and evaluated. The main contributions of this paper are the following:

1. The main requirements necessary for an encryption algorithm to be valid against the insider threat are presented.
2. A mechanism for the development of valid encryption algorithms to mitigate the insider threat is proposed.
3. A new encryption algorithm, with key generation mechanism based on contextual challenge execution, is presented to mitigate the insider threat.
4. A file marking mechanism is proposed that enables protection of the encryption algorithm against specific insider threat attacks.

The remainder of this article is organized as follows: The specificities and requirements of insider threat security are presented in section 2. In section 3, the SFS architecture and its component security techniques are described. Also, the Securecipher encryption algorithm is proposed in section 4, followed by the security and efficiency analysis of the proposed encryption algorithm in section 5. Finally, the main conclusions reached in this research are presented in section 6.

2. Insider threat security

This section outlines the primary specifications and requirements that must be considered for the development of an encryption algorithm within the context of insider threats.

2.1. Specificities of insider threats vs. external threats

In our study, we found that there is a significant difference between an attack derived from insider threats and one originating from external threats. An insider has certain “advantages” that an external attacker usually does not have. Table 1 presents the description of three specificities of an insider compared to an external attacker, studied in this research.

In Table 1, it is evident that due to the advantages presented by an insider, attacks known as known-plaintext attacks, chosen-plaintext attacks and chosen-ciphertext attacks can be produced (Kremer and Ryan, 2005; Lee,Lin,and Chang, 2011; Stamp and Low, 2007).

2.2. Requirements for designing an encryption algorithm resistant to insider threats

A valid encryption algorithm insider threats in SFS needs additional functionalities to those required by a conventional encryption algorithm. In this scenario, the user works with encrypted files in real time, which necessitates requests from applications at random positions. Table 2 outlines the requirements that an encryption algorithm adapted to insider threats must satisfy, classified into three types: functional, non-functional, and general. Functional requirements adapt the SFS in the context of insider threats. Non-functional requirements refer to the main considerations during the design of the algorithm. General requirements cover criteria applicable to all encryption algorithms.

3. Secure file system architecture

The underpinning of the proposed encryption algorithm’s research lies in the SFS, as shown in Fig. 1. The SFS is essentially a Virtual File System (VFS) enhanced with security methodologies. The Dokan tool has been employed for the development of the VFS (Anon, n.d.; Montano et al., 2022). Dokan is a library utilized for developing user-mode file systems on the Windows platform. This tool facilitates the creation of file systems that mirror the real file system. The VFS intercepts application calls to the operating system to implement security measures among them. The primary objective of the SFS, is to safeguard information by encrypting it upon exit from the VFS. The VFS intercepting write operations to external devices and reads from cloud storage repositories. Fig. 1 illustrates the architecture and operation of the SFS. Fig. 1 depicting a plaintext file that, upon interaction with the SFS, undergoes

Table 1
Specificities of insider threats vs. external threats. Own source.

Specificities	Insider threats	External threats
File generation	An insider has the capability to produce encrypted files, thereby acquiring as many as required.	The external attacker may capture information. However, unlike the insider, they cannot generate it.
Knowledge of the information	The insider can generate the files with the information he wants. This allows you to create files that can have very useful information depending on the encryption. Either files of a specific size (1 byte, 512 bytes, etc.), or files with a specific content (all bytes at 0x00, 0xFF, etc.).	An external attacker generally does not know the content of the captured information.
Correspondence of encrypted and decrypted files	The insider may know the correspondence between plaintext and encrypted files.	The external attacker has no access to knowledge of this correspondence.

Table 2
Requirements for designing an encryption algorithm resistant to insider threats. Own source.

Type	Requirement	Description
Functional	**Maintain message or file size**	An application can read the size of a file while it is in use, this requirement is necessary to avoid operating errors.
	Maintain the position of each byte	An application can make requests to different positions in a file under certain circumstances. This requirement ensures that applications will find the byte in the correct location.
	Encrypting and decrypting with different operations	Encrypting and decrypting must be distinct operations to prevent successive encryption processes from reversing the encryption.
	Decryption without encryption is valid	Decryption should be possible even without prior encryption so if you want to share a file with a client using an application that encrypts (such as emails), once it is encrypted with the keys generated by the client the file becomes clear. In this way, system overloads are avoided by unnecessarily encrypting and decrypting.
Non-Functional	**Lightweight Encryption**	In a SFS scenario, real-time operations demand encryption algorithms characterized by low computational overhead to minimize disruption of the file system's regular functionality.
	Statistically Robust	Ensure protection against external attacks.
	Robust against insider attack scenario	Ensure protection against insider attacks.
General	**Minimal encryption and decryption time**	Minimum time required to encrypt/decrypt in real time.
	Minimum memory usage	The performance of the cipher is influenced by factors such as the size of the key or initialization vectors, as well as the type of operations employed within the algorithm.
	Maximum throughput	Depending on the encryption time, it influences the power consumption of the algorithm.
	Avalanche effect	Determines whether a small change in the plaintext generates changes in the ciphertext.
	Optimal key size	This evaluation parameter defines the bandwidth required for transmission. It also helps to determine a size to avoid brute force attacks by ensuring that it is computationally impossible.
	Entropy	Entropy is used to measure randomness and uncertainty in data. The relationship between the ciphertext and the key becomes more complex with high randomness.

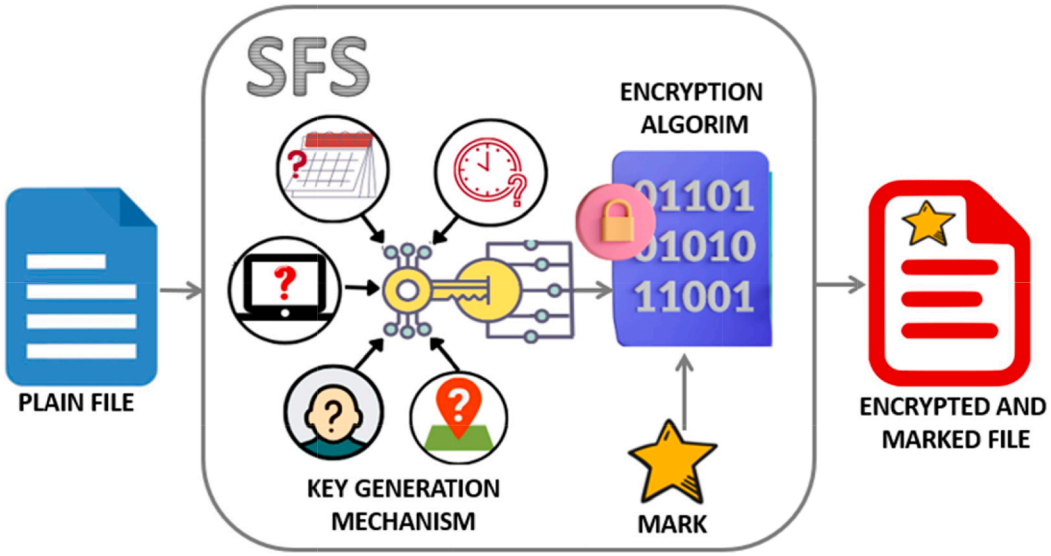


Fig. 1. SFS architecture and operation.

encryption and marking. The SFS comprises the VFS, a key generation mechanism based on challenge-solving, file marking mechanism, and the encryption system. Subsequently, the other components of the SFS are delineated.

3.1. Encryption algorithm

An encryption system refers to a procedure that uses an algorithm with a specific key, known as an encryption key. This procedure transforms a plain text message into a format unreadable to any individual who lacks the corresponding secret key, also known as the decryption key, associated with the algorithm (Faheem et al., 2017; Yegireddi and Kiran Kumar, 2016). The design of the encryption algorithm is based on the requirements previously studied and described in the previous section. The following is a description of how each requirement influences algorithm design:

Maintain message or file size: An encryption function that operates at the byte level is essential. This necessitates the use of a hash function that returns an element of size equivalent to the plaintext byte.

Maintain the position of each byte: The position has to influence the function, assuming that a byte is used as a clear element.

Encrypting and decrypting with different operations: The addition operation is proposed for encryption and the subtraction operation for decryption.

Decryption without encryption is valid: In this case, the byte in clear is subtracted from the result of the hash function.

Lightweight Encryption: Design of a lightweight cryptographic hash function.

Statistically Robus: Design of a cryptographic hash function that, starting from an input and a key, combines the bits in a diffuse and nonlinear way.

Robust against insider attack scenario: Inclusion of a File Random Number (*frn*) for each file, coming from the brand.

3.2. Hash function design mechanism

A hash function is a mathematical algorithm capable of transforming input data into a series of output characters, typically of a fixed length regardless of the input data's length. The output data represents a unique summary of all input information (Gupta,Goyal,and Aggarwal, 2014; Rivero-García et al., 2017). The diffusion in hash functions indicates that all input information influences the result, such that any alteration in the input data yields a different output. Hash functions, also referred to as one-way functions, exhibit confusion, establishing intricate relationships between input and output data. Consequently, deducing the input data solely from knowledge of the output becomes infeasible. To ensure the robustness of a hash function, it is imperative to uphold both effective diffusion and confusion properties (Dhanuja et al., 2020; Zhuoyu and Yongzhen, 2022).

The proposed encryption algorithm introduces a hash function that attempts to enhance the algorithm in both diffusion and confusion, as do well-known algorithms such as AES (Rijndael), DES, Serpent (AES finalist), Tiny Encryption Algorithm (TEA), among others (Joan and Vincent, 2002; Liu,Rijmen,and Leander, 2018; Sajadieh et al., 2012). For the development of the proposed hash function, boolean operations are studied. These operations allow high-speed byte-level processing and provide the cryptographic properties detailed in Table 3. With a well-balanced combination of the properties illustrated in Table 3, boolean functions are very advantageous in cryptographic applications (García, 2014).

According to (García, 2014), a boolean function cannot satisfy all properties at the same time. Functions with maximum nonlinearity, or functions of perfect nonlinearity have a reduced algebraic degree (maximum $n/2$, where n is the number of variables of the function) and are not balanced. The limit for the algebraic degree of resilient functions is $(n-m-1)$, where m is the order of resilience. The higher m denotes greater immunity to correlation, being balanced functions, but with non-linearity far from the maximum. That is why, for the study of boolean functions in this research, the software "Study of Boolean Functions" (Ramos Diaz 2022b) was developed, which generates boolean functions from their possible truth table, as shown in Fig. 2.

Fig. 2 presents the method of study of boolean functions used in this research. The cryptographic properties of the boolean functions resulting from a 4-variable truth table $(x1, x2, x3, x4$ are studied, in terms of balance, degree, resilience and nonlinearity. Then the boolean function with the best cryptographic properties is chosen.

3.3. Key generation mechanism

Conventional key generation mechanisms lack security against accidental leaks or theft, as they are typically stored (Ms. S. Suma et al., 2023). In this study, we introduce a key generation mechanism based on context challenge resolution. Specifically, a subkey is generated with the completion of each challenge. These challenges have various objectives, such as user identification, location, device, date, and time of accessing the information. The challenges' objectives can be resolved in different ways; for instance, user identification can be achieved through biometric or behavioral information. Biometric identification may involve facial recognition, voice recognition, fingerprints, etc., while behavioral identification can be through mouse dynamics, keystroke patterns, among others. Each objective has multiple methods of fulfillment, each generating a subkey. Concatenating these subkeys produces the encryption key, as illustrated in Fig. 3. This key generation method allows for key calculation rather than storage, enhancing their security. The generated key is of variable size. Subsequently, a key size setting process is performed to adjust it to a size of 8 bytes, as required by the encryption process. It can be seen in the Fig. 3, that in this key size setting process, it is checked if the key is greater or less than 8 bytes. If the key is larger, it is partitioned into subsets of 8 bytes and *XOR* operations are performed between the partitions. If, on the other hand, the key is smaller, a concatenation of the key is performed until it reaches a size of 8 bytes.

Table 3
Cryptographic properties of boolean operations in Securecipher. Own source.

Cryptographic property	Description
Non-linearity	Distancing of related functions. Prevents cryptographic attacks, making linear cryptanalysis more difficult.
Degree	Degree of the multivariate polynomial that represents the boolean function, a high degree implies more linear complexity.
Balance	Presence of the same number of ones and zeros in its truth table. This avoids statistical dependence between plaintext and ciphertext.
Resilience	Immunity to correlation between the values of the function and the previous knowledge of the binary values of some of its variables.

3.4. Marking mechanism

The mark is a mechanism incorporated into the SFS. This mechanism is necessary to determine the encryption or decryption action to be performed. It also allows the encryption algorithm to be protected against file generation attacks by insider threats. The proposed marking mechanism is incorporated into the file without changing its size, using the Huffman Code (Huffman, 1952). The Fig. 4 shows the structure of the marking and Table 4. describes each parameter that composes it.

The file marking and unmarking procedure is performed before the corresponding encryption or decryption action. Algorithms 1 and 2 describe the marking and unmarking procedure, respectively.

Algorithm 1 Marking Algorithm.

Marking Process
1. **Read the first 512 bytes of plain text**
plain_bytes = read plain_text [0:512]
2. **Bytes of plain text are compressed with Huffman Code**
Full_Compression_Result = encoding_Huffman (plain_bytes)
3. **The size of the compressed text is obtained**
FCS = size (Full_Compression_Result)
4. **Check that there is enough space to add the *frn* and level parameters (5 bytes). If this condition is true, the compressed text, filling sequence, *frn* and level are concatenated. Then the level is returned. If the condition is false, the message "INVALID_MARK_LEVEL" is returned.**
if (512-FCS)>=5:
plain = Concatenate(Full_Compression_Result, Filing_Sequence, *frn*, LVL)
return LVL
else:
INVALID_MARK_LEVEL
End Process

Algorithm 2 Unmarking Algorithm.

Unmarking Process
1. **Verify that the following requirements are met:**
if $((ODS == MARK\_LENGTH == 512)$ **AND** $(FCS <== (MARK\_LENGTH - 5))$ **AND** $(plain[FCS : (MARK\_LENGTH - 5)] == Filing\_Sequence)$ **AND** $(frn! = INVALID\_FRN)$ $AND(LVL == 1 OR LVL == -1))$:
$plain\_bytes = decoding\_Huffman(Full\_Compression\_Result)$
return frn, LVL
else:
return 0
End Process

The proposed marking mechanism has two known limitations. First, files smaller than 512 bytes are not marked and therefore are not encrypted or decrypted. This happens because the mechanism we propose needs at least 512 bytes to incorporate the compressed information, decompression, *frn* and level parameters. Secondly, it may be the case that the file, even if it is large enough to be marked, the compression is not enough to incorporate the decompression, *frn* and level parameters. This

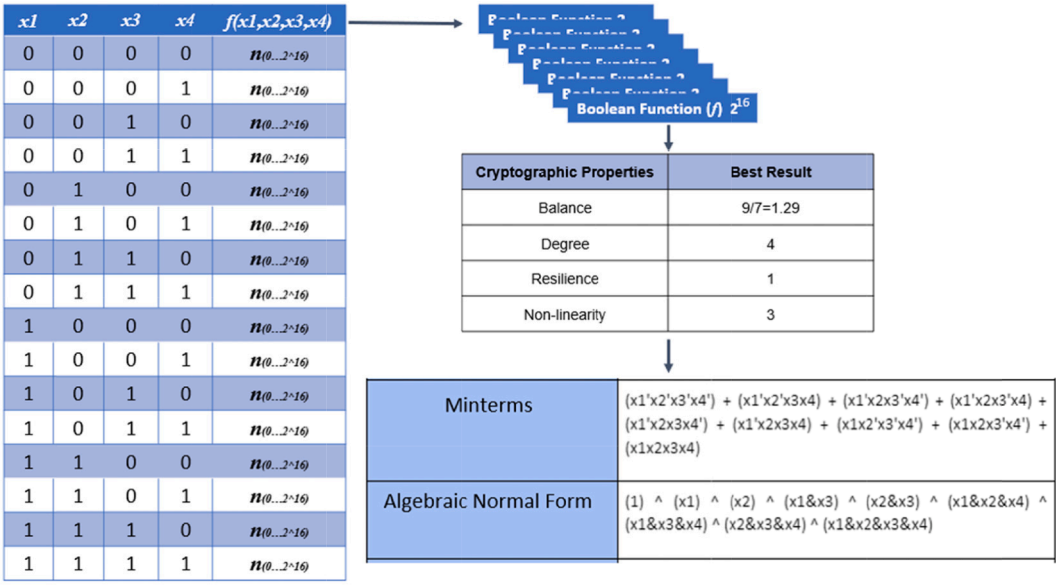


Fig. 2. Boolean function study method.

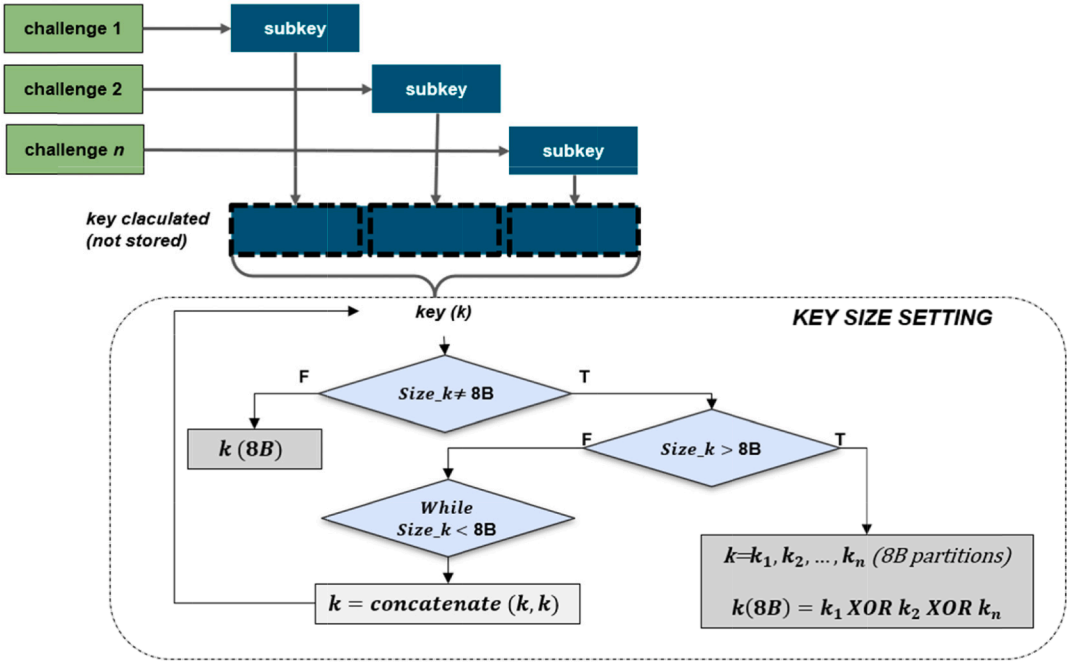


Fig. 3. Operation of the key generation mechanism and key size setting, (B = bytes).

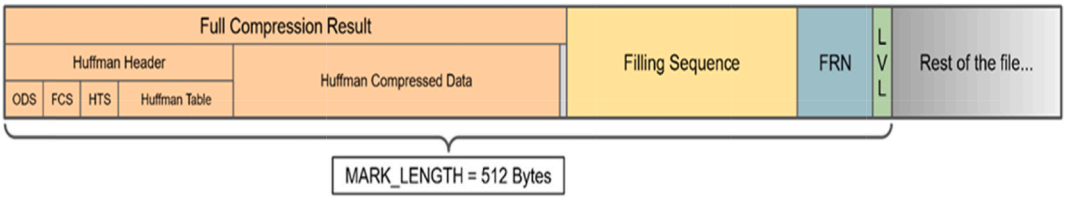


Fig. 4. Marking mechanism structure.

limitation derives from the fact that information is being added to the file without changing its length, i.e., inherent to tagging. In these cases, we have decided to block the output of these files from the computer.

4. Securecipher encryption proposal

The encryption algorithm proposed in our study is called Secure-

Table 4
Description of mark structure.

Components	Description	Size (bytes)
Full Compression Result	Information necessary to recover the original data and has a variable size.	*FCS*
Huffman Header	Information necessary to recover the original data and has a variable size.	*FCS*
Original Data Size (*ODS*)	Indicates the size in bytes of the original data and that have been replaced by the mark. Must always be MARK_LENGTH ==512.	2
Full Compression Size (*FCS*)	Indicates the size in bytes of the complete result (header included) of the compression of the original data. Size:	2
Huffman Table Size (*HTS*)	Indicates the size in bits of the Huffman table.	2
Huffman Table	The translations of the original bytes to encoded bytes necessary to be able to reverse the compression.	$\lceil HTS/8 \rceil$
Huffman Compressed Data	The compressed data, i.e. the original data encoded according to the Huffman table.	$FCS - 6 - \lceil HTS/8 \rceil$
Padding.	This padding only exists if the last byte that has been filled with the compressed data has not been completed, occupying up to 7 bits, so that the padding sequence is always aligned by bytes.	< 1
Filling Sequence	This is a predefined sequence with which the remaining space is filled and therefore has a variable size.	$507 - FCS$
File Random Number (*frn*)	Random number specific to this file.	4
Level (*LVL*)	Indicates the encryption level of the file. Only values −1 (decrypted) and 1 (encrypted) are valid.	1

cipher. Securecipher is an instantaneous synchronization stream cipher algorithm focused on the requirements needed to help mitigate the insider threat. Fig. 5 shows the operation of the encryption algorithm and depicts the size of each parameter. From the original file the byte in plain (pB) is taken and added to the byte resulting from the function f to encrypt. As well as the encrypted byte cB((from an encrypted file) is taken and subtracted from the byte resulting from the function f to decrypt. Equations (1) and (2) represent the encryption and decryption equations of the proposed algorithm, respectively.

$$cB = pB + f(p, frn, k) \tag{1}$$

$$dB = cB - f(p, frn, k) \tag{2}$$

Where, cB is the encrypted byte, pB is the plaintext byte, f is the hash function, k is the key generated by the context-based key generation mechanism, p is the position in the text of the plaintext byte being encrypted or decrypted, frn is the random number obtained from the marking mechanism and dB is the decrypted byte. The key generated by solving the contextual challenges is the same for all encrypted files, but the frn is random and unique for each file.

The parameters p, frn, k are concatenated to create the input message to the hash function, as shown in Fig. 5. The lightest possible diffusion and confusion mechanisms have been used for the design of f. To provide diffusion, the input message is split into two vectors and a linear transformation is performed, as shown in equations (3) and (4).

$$c = a + b \tag{3}$$

$$d = c + 2{*}b \tag{4}$$

Where a and b are the vectors into which the input message has been divided, c and d are the new vectors created from vectors a and b. For the confusion, a study of the truth table of boolean operations with 4 variables was performed, as described above. From the study, equation (5) was obtained as the best result.

$$f = (t) \wedge (x) \wedge (t\&y) \wedge (x\&y) \wedge (t\&x\&z) \wedge (t\&y\&z) \wedge (x\&y\&z) \wedge (t\&x\&y\&z) \tag{5}$$

Where t, x, y, z are the 5 bytes fragments into which the message is divided. The encryption and decryption algorithms in Algorithms 3 and 4 are shown below, respectively.

Algorithm 3 Encryption Algorithm.

Encryption Process
1. Read pB, p, frn, k, of fixed sizes of $1, 8, 4 and 8$ bytes respectively.
2. Concatenate p, frn, k to create message (20bytes)
3. Split the message into substrings of 10 bytes to create variables a and b.
4. Create variables c and d from a and b as shown in equations (3) and (4) respectively.
5. Create substrings t and x of 5 bytes each from variable c.
6. Create the substrings y and z of 5 bytes each from variable d.
7. Compute the hash function as shown in equation (5).
8. Compute the encrypted byte as shown in equation (1).

End Process

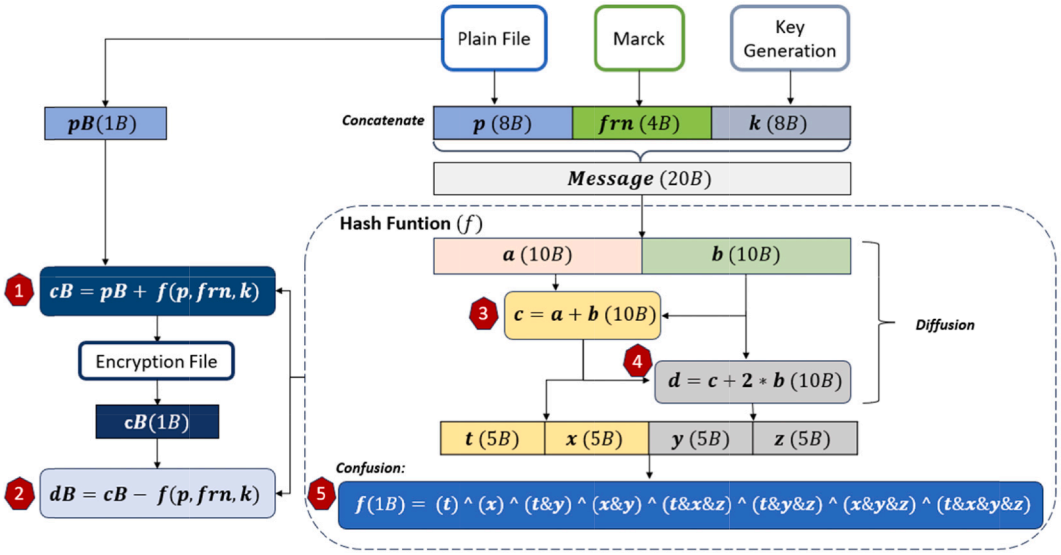


Fig. 5. Operation of the encryption algorithm, (B = bytes).

Algorithm 4 Decryption Algorithm.

Decryption Process
1. Read pB, p, frn, k, of fixed sizes of $1, 8, 4 and 8$ bytes respectively.
2. Concatenate p, frn, k to create message (20bytes)
3. Split the message into substrings of 10 bytes to create variables a and b.
4. Create variables c and d from a and b as shown in equations (3) and (4) respectively.
5. Create substrings t and x of 5 bytes each from variable c.
6. Create the substrings y and z of 5 bytes each from variable d.
7. Compute the hash function as shown in equation (5).
8. Compute the encrypted byte as shown in equation (2).

End Process

5. Validation

Validation of the proposed encryption algorithm is performed by validating the diffusion, confusion and performance analysis (Faheem et al., 2017). Diffusion is evaluated using software (Ramos Diaz 2022a) created to evaluate the influence of *frn* mechanism on the algorithm. The validation of the confusion is measured using the NIST statistical suite. The performance analysis is tested by calculating the entropy and the time required by the proposed algorithm for encryption and decryption. Moreover, a comparative analysis of our proposal and RC4 algorithm is performed in terms of entropy and performance.

5.1. Diffusion validation

In Securecipher, the main objective of the broadcast is to protect the encryption algorithm from the insider threat. For this purpose, the parameter *frn* is introduced to protect the encryption algorithm from specific insider threat attacks. To measure the influence of *frn* on the encryption algorithm, the following parameters were considered: plain files filled with zeros of size equal to 2; 8; 26and260bytes; 100 different of size equal to 8bytes encryption keys and 512 different *frn* values of size 9 bits. To analyze the influence of *frn* on the cipher, a specific program (Ramos Diaz 2022a) that performs the sequence of steps shown in Algorithm 5. In this way, the avalanche effect of our encryption algorithm is also analyzed.

Algorithm 5 Diffusion Validation Algorithm.

Diffusion Validation Process
1. Generate as many files as there are *frn* and key possibilities, as follows:
For $j = 0$ ***to*** *key_number*
For $i = 0$ ***to*** *frn_number*
While *size_generated_file* $<$ *size_plainfile*
$cB = Encryption(p, frn, k)$
2. Count the number of distinct files generated for each file size.
3. Calculate the number of expected files for each file size, as shown in the equation 6.
$number\_files\_expected = key\_number * frn\_number \quad (6)$
4. Calculate the Mean Square Error (MSE) between the number of expected files and the number of distinct files generated, as shown in the equation 7.
$$MSE = \frac{(number\_files\_distinct - number\_files\_expected)^2}{key\_number * frn\_number} \quad (7)$$
End Process

Where j and i are iterators; *key_number* is the number of keys to test with; *frn_number* is the number of *frns* to test with; *size_generated_file* and *size_plain_file* are the size of the generated file and the size of the key file, respectively; cB is the encrypted byte; p is the byte position in the blank file; *frn* is the random single parameter generated by flag, k is the key; *number_files_expected* is the number of different files expected and *number_files_distinct* is the number of different files generated.

The test has been carried out with small files, because the smaller the file size, the more difficult it is to generate different files. As shown in Fig. 6, the MSE is reduced in 8, 16 and 260 byte files with respect to the 2 byte file. In other words, the 2 byte file is less likely to change because of its smaller size. Consequently, the diffusion and avalanche effect tends to increase as the file size increases.

5.2. Confusion Validation

To statistically validate the encryption algorithm, the tests defined by NIST (National Institute of Standards and Technology) to statistically validate pseudo-random number generators are performed (Schneier, 1996). The NIST 800–22 test suite is a statistical package consisting of 15 tests for testing the randomness of binary (arbitrarily long) sequences produced by hardware or software cryptographic random or pseudo-random number generators (Rukhin et al., 2001). These tests focus on

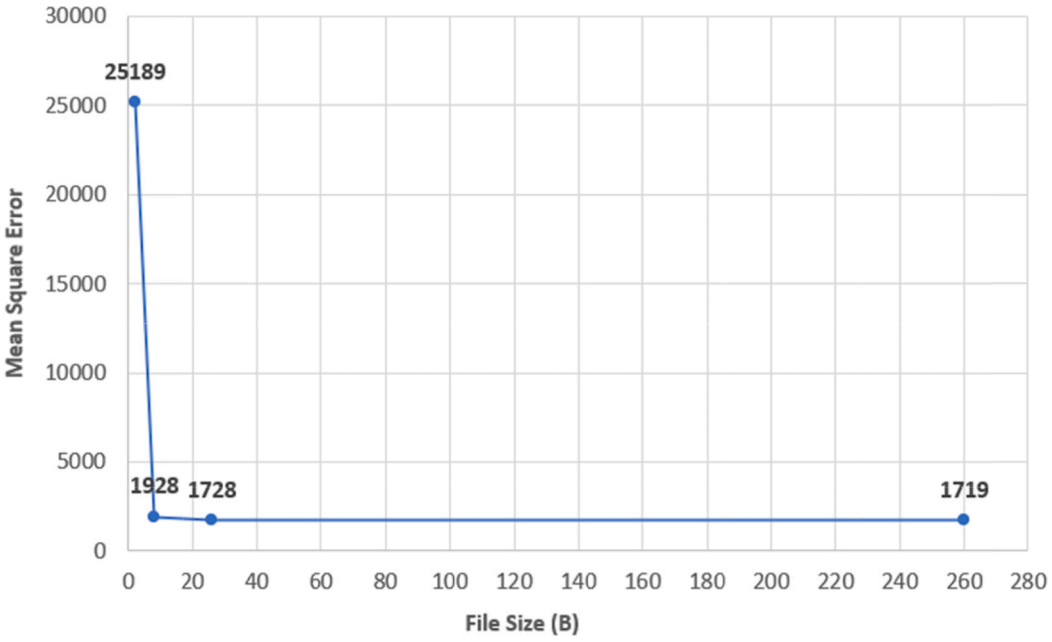


Fig. 6. Evolution of the MSE in files of size 2; 8; 16 and 260 bytes (B).

the different types of randomness that might exist in a sequence, and how it resembles what a sequence should really look like (Sýs et al., 2015).

The results of the NIST tests performed on Securecipher and the parameters used in each of them are shown in Table 5. Securecipher is considered to have successfully passed the NIST tests, despite failing two of them. Because according to (Sýs et al., 2015) a true pseudo-random number generator has an 80 % probability of failing at least one of the 15 tests. In the Overlapping Template Matching Test the input data does not match the test, which causes 119 subtests out of a total of 148.

5.3. Performance analysis

The performance analysis of the algorithm was performed by evaluating the time required for encryption and decryption, avalanche effect and entropy. Table 6 describes the tests performed in this study and their description.

To measure the encryption and decryption time of the proposed algorithm, a computer with Intel(R) Core(TM) i7-9850H CPU @ 2,60 GHz and 16 GB RAM was used. Fig. 7 shows the encryption and decryption time for 1 KB, 1 MB, 10 MB and 128 MB files. This test demonstrates that the encryption and decryption process occurs in a manner compatible with normal computer use. The encryption time of this test for a 10 MB text file is 0.54 s and the decryption time 0.52 s. In comparison with the study (Adeniyi et al., 2023), the results are significantly higher, considering that for an 8 MB file the encryption time is 7.2 s and the decryption time is 8.2 s, with the proposed Modified AES algorithm.

To measure the entropy of Securecipher, the frequencies of each word of the original file (Don Quixote Spanish version, size 128*MB*) are counted in binary, to obtain the vocabulary and the frequency of these. Then the same operation is performed for the file encrypted with Securecipher and compared. From the plaintext vocabulary 1.080 words were obtained and from the ciphertext vocabulary 45.152 words were obtained. Fig. 8 a and b show the frequencies of the words obtained in the vocabulary of each case. It is possible to observe an increase in the dictionary of distinct words, and a greater uniformity in the frequency of these once the original file has been encrypted. Although an optimal distribution of word frequency is not achieved.

5.4. Comparative analysis

During the study, cost and efficiency comparisons were made with the RC4 algorithm. The test was performed by calculating the encryption and decryption time of the Securecipher and RC4 algorithms; as well as the last byte access time for $0,001; 1; 10 and 128$MB files.

Table 7 shows that the RC4 algorithm is significantly faster than Securecipher. However, in an insider threat scenario, Securecipher is more effective than RC4 or any other known encryption algorithm because it allows instant access to the contents of any position in the file. In the specific case of RC4, due to its operation, it must encrypt or decrypt the entire file to access the last position, with a delay time of 0.25 s. Consequently, RC4, as well as any algorithm developed without considering the specificities of the insider threat scenario, has a delay time to access the last byte of a file longer than Securecipher.

Fig. 9 shows a comparison of the entropy behavior of the Don Quixote Spanish version file (size 128 MB) encrypted with Securecipher and RC4 algorithms. A vocabulary of 65,256 words was obtained with RC4, while with Securecipher only 45,152 words. Furthermore, as shown in the figure, RC4 has a much more uniform frequency than Securecipher, which is a limitation against cryptanalytic attacks. To counter this limitation, Securecipher has been complemented by the

Table 5
NIST 800–22 test results applied to Securecipher. Own source.

TEST	n	M	STREAMS	STATUS
Frequency (Monobit) Test	1000	–	*25*	✓
Frequency (Block) Test	2000	*20*	*15*	✓
Runs Test	1000	–	*15*	✓
Longest Run of Ones in a Block Test	2000	–	*15*	✓
Binary Matrix Rank Test	128	–	*15*	✓
Discrete Fourier Transform (Spectral) Test	2000	–	*10*	✓
Non-overlapping Template Matching Test	2000	–	*25*	✓
Overlapping Template Matching Test	20,000	*10*	*500*	*101*✓ *18*×
Maurer's Universal Statistical Test	6500	*9*	*25*	✓
Linear Complexity Test	–	–	–	×
Serial Test	2000	*3*	*10*	✓
Approximate Entropy Test	1,000,000	–	*1000*	✓
Cumulative Sums Test	1,000,000	–	*1000*	✓
Random Excursions Test	2000	*8*	*20*	✓
Random Excursions Variant Test	1,000,000	*500*	*25*	✓

Table 6
Description of the Securecipher performance analysis. Own source.

Test	Description
Time required for encryption and decryption	Encryption time required to adapt the cipher to the desired context (a very slow cipher is not suitable for real-time use).
Avalanche effect	It determines whether a small change in the plaintext generates large changes in the output. In the case of Securecipher, depending on the position, the avalanche effect is diminished, but thanks to the *frn* mechanism, this effect is maintained automatically since it changes the input of the hash function at each modification, and therefore the result of the cryptographic hash, allowing not only the position byte to change, but all the bytes of the file. This property is validated in the diffusion validation study.
Entropy	Entropy is used to measure randomness and uncertainty in the data. The relationship between the ciphertext and the key becomes more complex with high randomness. Entropy is measured by counting the frequencies of each word in the original file in binary, to obtain the "vocabulary", i.e. the number of hexadecimal words used and their frequency. The more the vocabulary grows in the encrypted file with respect to the clear one and the more similar their frequency is, the higher the entropy will be in the encrypted file.

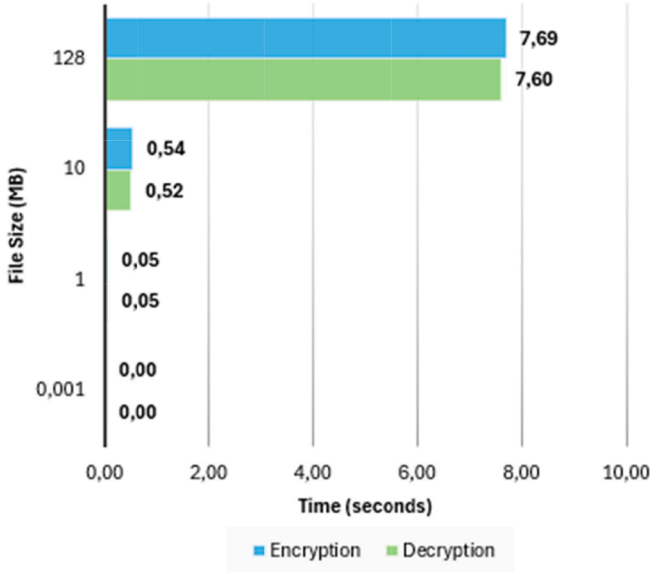


Fig. 7. Encryption and decryption time of $0,001; 1; 10 and 128 megabite (MB)$ files with Securecipher.

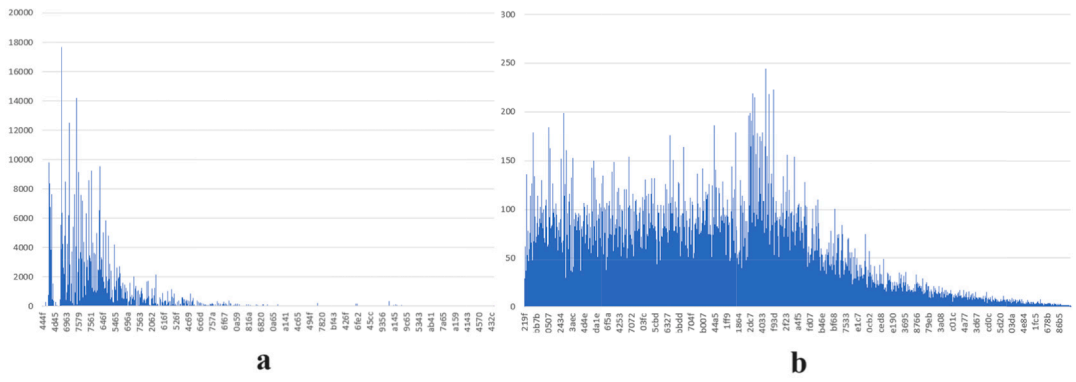


Fig. 8. A) entropy of plaintext. b) entropy of ciphertext with securecipher.

Table 7
Comparison of the performance analysis and time required to access the last byte of the file with Securecipher and RC4.

File Size	Securecipher Encryption	Decryption	RC4 Encryption/ Decryption	Last Byte Access Securecipher	RC4
1 KB	–	–	–	–	–
1 MB	0,052	0,052	0.002	–	0.002
10 MB	0,542	0,515	0,018	–	0.018
128 MB	7,69	7,6	0.25	–	0.25

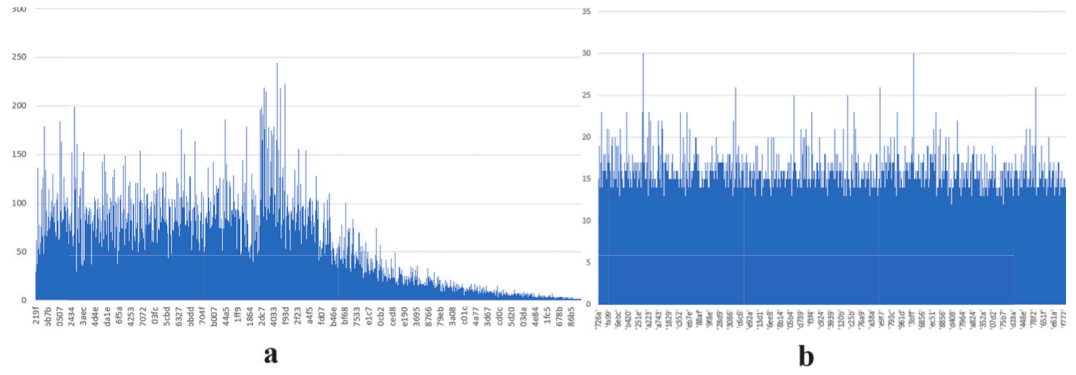


Fig. 9. A) entropy of the ciphertext with securecipher. b) entropy of the ciphertext with rc4.

FRN parameter introduced by the marking mechanism. Thus, each file the user creates will generate a different FRN result and thus a different encrypted file, even if the file in clear is the same.

6. Conclusions

In this paper, we propose a new encryption algorithm for SFS against the insider threat, and the requirements that an algorithm must meet for this purpose. A mechanism for the development of this type of encryption algorithms is described and a file marking mechanism is presented. In addition, a software that allows the study of boolean functions is proposed, and another software for the advanced study of the diffusion of this type of algorithm.

Starting from the requirements imposed by the DLP use case, an algorithmic definition of a novel cipher has been arrived at. The proposed encryption algorithm passes 87 % of the NIST tests successfully, equivalent to 13 successful tests out of 15. It has been proven that for the insider threat use case the proposed algorithm is more efficient than the RC4 encryption algorithm, since it is faster when accessing a given file position. It was further proved that the incorporation of the document marking mechanism is necessary in a valid algorithm for SFS against the insider threat. The proposed marking mechanism, despite having some limitations in applying the security techniques to files smaller than 512 bytes in size, allows increasing the diffusion and security of the algorithm. Although an increase in the vocabulary of Securecipher ciphertext with respect to natural language of approximately 42 times and in the stability of word frequencies has been demonstrated, the RC4 algorithm was found to have more stability in frequencies and a larger number of words in the vocabulary. This means that the confusion of our algorithm presents limitations in the face of cryptanalysis attacks. However, this limitation has been countered by the inclusion of the FRN parameter that makes each encrypted file different, even if the file in clear is the same. For this reason, we consider Securecipher to be valid for insider threat scenarios, as it meets the design criteria required in this type of scenario and has a high diffusion.

Therefore, as future work, we propose to continue investigating this type of encryption algorithms to obtain an optimal algorithm, both for insider and external threats. Also, we intend to address the limitations of the tagging mechanism so that users can share any type of files with built-in security techniques.

7. Original article statement

This manuscript is the authors' original work and has not been published or has it been submitted simultaneously elsewhere.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

This research has been partially supported by the Center for the Development of Industrial Technology (CDTI), Spanish Ministry of Science and Innovation under the project named "Secureworld: Technologies for secure digital relationships in a hyperconnected world", IDI-20200518. Isabel Herrera Montano has been funded through the UVa 2022 predoctoral contracts call, co-financed by Santander Bank.

References

Abujoodeh, Mohammed, Liana Tamimi, and Radwan Tahboub. 2023. "Toward Lightweight Cryptography: A Survey." in *Computational Semantics*. IntechOpen.

Adeniyi, A. E., Abiodun, K. M., Awotunde, J. B., Olagunju, M., Ojo, O. S., & Edet, N. P. (2023). Implementation of a Block Cipher Algorithm for Medical Information Security on Cloud Environment: Using Modified Advanced Encryption Standard Approach. *Multimedia Tools and Applications, 82*(13), 20537–20551. https://doi.org/10.1007/s11042-023-14338-9

Ahmad, S., Mehfuz, S., & Beg, J. (2022). "Cloud Security Framework and Key Management Services Collectively for Implementing DLP and IRM". *Materials Today Proceedings*. https://doi.org/10.1016/j.matpr.2022.03.420

Ahmad, S., Mehfuz, S., & Beg, J. (2023). Hybrid Cryptographic Approach to Enhance the Mode of Key Management System in Cloud Environment. *The Journal of Supercomputing, 79*(7), 7377–7413. https://doi.org/10.1007/s11227-022-04964-9

Anon. n.d. "Dokan." Retrieved (https://github.com/dokan-dev/dokany/wiki/Build).

Bandari, V. (2021). Enterprise Data Security Measures: A Comparative Review of Effectiveness and Risks Across Different Industries and Organization Types. *International Journal of Business Intelligence and Big Data Analytics, 6*, 1–11.

Bernstein, D. J. (2024). Cryptographic Competitions. *Journal of Cryptology, 37*(1), 7. https://doi.org/10.1007/s00145-023-09467-1

v. Bhagat, Priti, Kaustubh S. Satpute, and Vikas R. Palekar. 2013. "Reverse Encryption Algorithm: A Technique for Encryption \& Decryption." *International Journal of Latest Trends in Engineering and Technology (IJLTET)* 2:90–95.

Bhondve, Mrs. Pooj. Sameer. 2023. "Efficiently Encryption Decryption Schema for Secure File Storage System in Cloud Computing." *Interantional Journal Of Scientific Research In Engineering And Management* 07(07). doi: 10.55041/IJSREM25005.

Burnett, S., & Paine, S. (2001). *RSA Security's Official Guide to Cryptography*. USA: McGraw-Hill Inc.

Dhanuja, B., Prabadevi, B., Bhavani Shankari, K., & Sathiya, G. (2020). E-REA Symmetric Key Cryptographic Technique. In *in 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). IEEE* (pp. 1–8).

Djordjevic, I. B. (2019). Conventional Cryptography Fundamentals. In *Physical-Layer Security and Quantum Key Distribution* (pp. 65–91). Cham: Springer International Publishing.

Dworkin, Morris J. 2023. *Advanced Encryption Standard (AES)*.

Faheem, M., Jamel, S., Hassan, A., Zahraddeen, A., Shafinaz, N., & Mat, M. (2017). A Survey on the Cryptographic Encryption Algorithms. *International Journal of Advanced Computer Science and Applications, 8*(11). https://doi.org/10.14569/IJACSA.2017.081141

García, F. J. (2014). *Soporte, Grado y No Linealidad Perfecta de Funciones Booleanas*. Universitat d'Alacant/Universidad de Alicante.

Ghouse, M., Nene, M. J., & VembuSelvi, C. (2019). Data Leakage Prevention for Data in Transit Using Artificial Intelligence and Encryption Techniques. In *in 2019 International Conference on Advances in Computing, Communication and Control (ICAC3). IEEE* (pp. 1–6).

Gudes, E. (1980). The Design of a Cryptography Based Secure File System. *IEEE Transactions on Software Engineering SE-6(5):411–20.*. https://doi.org/10.1109/TSE.1980.230489

Gupta, S., Goyal, N., & Aggarwal, K. (2014). A Review of Comparative Study of MD5 and SSH Security Algorithm. *International Journal of Computer Applications, 104*(14), 1–4. https://doi.org/10.5120/18267-9305

Herrera Montano, Isabel, José Javier García Aranda, Juan Ramos Diaz, Sergio Molina Cardín, Isabel de la Torre Díez, and Joel J. P. C. Rodrigues. 2022. "Survey of Techniques on Data Leakage Protection and Methods to Address the Insider Threat." *Cluster Computing* 25(6):4289–4302. doi: 10.1007/s10586-022-03668-2.

Huang, Q., & Li, H. (2017). An Efficient Public-Key Searchable Encryption Scheme Secure against inside Keyword Guessing Attacks. *Information Sciences, 403–404*, 1–14. https://doi.org/10.1016/j.ins.2017.03.038

Huffman, D. (1952). A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE, 40*(9), 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898

Joan, D., & Vincent, R. (2002). The Design of Rijndael: AES-the Advanced Encryption Standard. *Information Security and.* Cryptography.

Kremer, S., & Ryan, M. D. (2005). Analysing the Vulnerability of Protocols to Produce Known-Pair and Chosen-Text Attacks. *Electronic Notes in Theoretical Computer Science, 128*(5), 87–104. https://doi.org/10.1016/j.entcs.2004.11.043

Lee, C.-C., Lin, T.-H., & Chang, R.-X. (2011). A Secure Dynamic ID Based Remote User Authentication Scheme for Multi-Server Environment Using Smart Cards. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2011.04.190

Liu, Y., Rijmen, V., & Leander, G. (2018). Nonlinear Diffusion Layers. *Designs, Codes and Cryptography, 86*(11), 2469–2484. https://doi.org/10.1007/s10623-018-0458-5

Montano, Isabel Herrera, Isabel de La Torre Diez, Jose Javier Garcia Aranda, Juan Ramos Diaz, Sergio Molina Cardin, and Juan Jose Guerrero Lopez. 2022. "Secure File Systems for the Development of a Data Leak Protection (DLP) Tool Against Internal Threats." pp. 1–7 in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE.

Ms, S., Suma, S. F., Rahuman, A. G., & Hari Ganesh, R. (2023). File Storage System Using Hybrid Cryptography. *International Journal of Advanced Research in Science, Communication and Technology, 45–49*. https://doi.org/10.48175/IJARSCT-8659

Nie, Tingyuan, and Teng Zhang. 2009. "A Study of DES and Blowfish Encryption Algorithm." Pp. 1–4 in *TENCON 2009 - 2009 IEEE Region 10 Conference*. IEEE.

Pal, P., Chattopadhyay, P., & Swarnkar, M. (2023). Temporal Feature Aggregation with Attention for Insider Threat Detection from Activity Logs. *Expert Systems with Applications, 224*, Article 119925. https://doi.org/10.1016/j.eswa.2023.119925

Paul, B. (2018). Steganography in XML Files Using RC4 Stream Encryption Algorithm. *International Journal for Research in Applied Science and Engineering Technology, 6*(4), 4727–4731. https://doi.org/10.22214/ijraset.2018.4776

Ramos Diaz, Juan. 2022a. "Securecipher Difussion Validation.".

Ramos Diaz, Juan. 2022b. "Study of Boolean Functions.".

Renaud, K., Warkentin, M., Pogrebna, G., & van der Schyff, K. (2024). VISTA: An Inclusive Insider Threat Taxonomy, with Mitigation Strategies. *Information & Management, 61*(1), Article 103877. https://doi.org/10.1016/j.im.2023.103877

Rivero-García, Alexandra, Iván Santos-González, Candelaria Hernández-Goya, and Pino Caballero-Gil. 2017. "IBSC System for Victims Management in Emergency Scenarios." Pp. 276–83 in *IoTBDS 2017 - Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*.

Rukhin, Andrew, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. 2001. "NIST Special Publication 800-22." 22.

Sajadieh, Mahdi, Mohammad Dakhilalian, Hamid Mala, and Pouyan Sepehrdad. 2012. "Recursive Diffusion Layers for Block Ciphers and Hash Functions." Pp. 385–401 in.

Schneier, Bruce. 1996. *Applied Cryptography Protocols*.

Smid, M. E., & Branstad, D. K. (1988). Data Encryption Standard: Past and Future. *Proceedings of the IEEE, 76*(5), 550–559. https://doi.org/10.1109/5.4441

Stallings, William. 2005. "The RC4 Stream Encryption Algorithm.".

Stamp, M., & Low, R. M. (2007). *Applied Cryptanalysis: Breaking Ciphers in the Real World*. Wiley.

Sýs, M., Říha, Z., Matyáš, V., Márton, K., & Suciu, A. (2015). On the Interpretation of Results from the NIST Statistical Test Suite. *Romanian Journal of Information Science and Technology, 18*(1), 18–32.

Thakare, R. D., Suryawanshi, Y., Jain, S., Bhagat, A. R., Patil, P. T., & Chore, N. M. (2024). Analysis of RSA Cryptosystem to Secure Messages in Vehicular Adhoc Network. *International Journal of Intelligent Systems and Applications in Engineering, 12* (10s), 360–1338.

Tihanyi, N. (2022). Report on the First DES Fixed Points for Non-Weak Keys: Case-Study of Hacking an IoT Environment. *IEEE Access, 10*, 77802–77809. https://doi.org/10.1109/ACCESS.2022.3192399

Yegireddi, R., & Kiran Kumar, R. (2016). A Survey on Conventional Encryption Algorithms of Cryptography. In *in 2016 International Conference on ICT in Business Industry & Government (ICTBIG)* (pp. 1–4). IEEE.

Zhao, J. (2023). DES-Co-RSA: A Hybrid Encryption Algorithm Based on DES and RSA. In *in 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA)* (pp. 846–850). IEEE.

Zhuoyu, H., & Yongzhen, L.i. (2022). Design and Implementation of Efficient Hash Functions. In *in 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)* (pp. 1240–11123). IEEE.