International Journal of Safety and Security Engineering
Vol. 14, No. 1, February, 2024, pp. 47-61
Journal homepage: http://iieta.org/journals/ijsse

Enhancing Data Security in Multi-Cloud Environments: A Product Cipher-Based Distributed Steganography Approach

Syed Shakeel Hashmi[1*], Arshad Ahmad Khan Mohammad[2], Arif Mohammad Abdul[2], Choudapur Atheeq[2], Mohammad Khaja Nizamuddin[2]

[1] Department of Electronics and Communication Engineering, Faculty of Science and Technology (IcfaiTech), the ICFAI Foundation for Higher Education (Deemed to be University), Hyderabad 501203, India
[2] Department of Computer Science and Engineering, School of Technology, GITAM (Deemed to be University), Hyderabad 502329, India

Corresponding Author Email: hashmi@ifheindia.org

https://doi.org/10.18280/ijsse.140105

Received: 21 September 2023
Revised: 26 December 2023
Accepted: 16 January 2024
Available online: 29 February 2024

Keywords:
multi-cloud computing, data security, steganography, Product Cipher-Based Distributed Steganography (PCDS)

ABSTRACT

In multi-cloud computing, securing sensitive data remains a paramount challenge. This paper presents a novel steganographic methodology, Product Cipher-Based Distributed Steganography (PCDS), designed to securely hide data within a multi-cloud environment. This approach, addressing the intricacies of decentralized data concealment, utilizes unaltered cover media as benchmarks for fragmenting and disguising data. The PCDS scheme, by distributing hidden data dynamically across multiple cloud platforms, successfully evades detection through the absence of file modifications or the use of special characters. An in-depth security analysis of this method demonstrates its resilience against unauthorized access; even with complete access to all cloud accounts involved, the extraction of the concealed message remains computationally unfeasible. The utilization of an undisclosed key, alongside a base encoding value and the inherent computational complexity of the scheme, fortifies its defense against brute-force attacks, significantly elevating its security profile compared to existing methods. This paper contributes substantially to the field of cloud security and steganography by offering an undetectable and innovative approach for data hiding. It effectively counters prevailing vulnerabilities in multi-cloud storage and sets a new precedent for advanced secure data concealment strategies. Contrasting with conventional methods susceptible to brute-force attacks requiring substantially fewer computations, the PCDS framework ensures a higher level of security, providing robust protection for confidential data in cloud environments.

1. INTRODUCTION

In the digital landscape of today's interconnected world, our reliance on the internet has grown exponentially, permeating diverse facets of our lives through online services. However, this widespread interconnectivity has exposed us to heightened security risks, compelling the development of innovative tools to protect sensitive information. In response, the synergy of cryptography and steganography has emerged as a formidable defense. Cryptography involves the intricate transformation of data into an unreadable format using complex algorithms, while steganography conceals information within other data, reducing its detectability.

The enduring significance of cryptography in ensuring data confidentiality and integrity is undeniable. Nevertheless, steganography offers a captivating approach by obfuscating hidden data and embedding confidential messages within innocuous cover messages guided by cryptographic keys. The essence of steganography lies in the formidable challenge it poses to unauthorized entities attempting to retrieve hidden messages without the proper key, adding an additional layer of security [1-7].

The potency of steganography hinges on several critical factors, encompassing the effectiveness of message concealment within cover data, the data's capacity to seamlessly host concealed messages, and the technique's resilience against potential adversarial attacks. The selection of an appropriate carrier medium is paramount in steganography. The ubiquity, transparency, and accessibility of cloud computing position it as an optimal candidate for this role [8].

Cloud computing, a cornerstone of distributed computing, provides access to versatile computational resources, boasting advantages such as cost-efficiency, enhanced data accessibility, and accelerated computational speeds [9-14]. Despite these benefits, the paramount challenge of ensuring data security within cloud environments persists. To confront this challenge, the amalgamation of cryptography and steganography has emerged as a robust approach, enhancing the security of sensitive information and mitigating potential threats [15-17].

Steganography's operation involves covertly embedding

crucial information within different data types, such as images or audio files. This covert process ensures that even if visible data undergoes minor alterations, hidden information remains secure and retrievable. The steganography realm can be broadly categorized into two primary classifications: classical and distributed steganography [18]. Classical steganography conceals data within a single cover medium, while distributed steganography fragments and disseminates data across multiple mediums, substantially amplifying the complexity of detecting or accessing the concealed message.

In the historical context of classical steganography, an intriguing analogy was introduced by Simmons in 1984—the "prisoner's problem." This analogy depicts a scenario where two individuals, Alice and Bob, communicate secretly under close surveillance [19]. This metaphor underscores the intricate challenge of establishing covert communication channels that evade detection. Classical steganography employs sophisticated techniques to conceal confidential messages within various media types, leveraging Secret keys for clever insertion and extraction. The primary objective is utmost discretion, thwarting potential adversaries from deciphering the process and exposing the messages. However, as adversaries become more sophisticated and capable of detecting hidden messages and meticulously analyzing exchanged data, the necessity to establish robust defenses intensifies. This interplay between secrecy and detection holds critical importance, particularly in the context of cloud computing, where secure information exchange fuels the evolution of steganography techniques. Advanced statistical tools are employed to reveal intricate patterns of data concealment and revelation in the digital realm [20, 21].

The realm of distributed steganography introduces a novel paradigm, building on classical steganography's foundations [22-26]. This approach involves fragmenting sensitive messages into discreet segments, which are then distributed across an array of covert communication channels. This strategy adds a layer of complexity, significantly enhancing the challenge of detecting and reconstructing the concealed message [27-32]. This technique finds utility in scenarios where multiple independent parties collaborate covertly. For instance, Liao et al. [33] introduced a model wherein different participants possess distinct covert messages, with the intended recipient having all hidden messages to reconstruct the original content. This Secret-sharing strategy entails three main steps: creating a target key, distributing share keys, and ultimately reconstructing the Secret. The delicate balance between security and accessibility is at the core of this process.

Distributed steganography's security is fortified by distributing the Secret across various media types. However, introducing subtle changes to accommodate a hidden message may inadvertently arouse suspicion, exposing the risk of steganalysis [34-36]. Amid these complexities, the potential for complete loss of secrecy due to media tampering necessitates careful consideration. The challenge in distributed steganography is striking the right balance between complexity for safety and durability in a dynamically changing digital realm.

To address limitations in traditional steganography, a new concept has been proposed where the cover media remains unchanged and serves as a pointer to fragmented data stored in a multi-cloud environment. This approach makes locating and extracting the Secret message difficult for attackers. Leonel Moyou Metcheka and Ndoundam [37] employ steganography to protect the password. It is claimed that the existing covert channel's new steganography idea makes it difficult for an attacker to figure out how to extract the hidden message. Two technical contributions served as the foundation for the current system's architecture. First, the cover media are not altered; rather, they function as a link to fragmented data. Second, the multi-cloud storage environment contains a Secret message. Cover media is chosen and uploaded into the cloud based on the message that communicating entities desire to keep hidden. Here, the method uses files as the cover media and uploads them directly into the cloud. Using the Secret message and key will allow you to upload the files. The key includes the following details: number of clouds, in the following order: C_0, C_1, …… C_{n-1} and their login information, file list, such as L^0, L^1, …… L^{k-1}, each list consisting of a set of files, such as ($L^0{}_0, L^0{}_1, \ldots\ldots L^0{}_{B-1}$) are composed in L^0 and ($L^1{}_0, L^1{}_1, \ldots\ldots L^1{}_{B-1}$) are composed in L^1 and so on with the encoding base values. Work presupposed that the communication entities would safely share the key information.

To secretly store the Secret in the cloud, the sender transcodes the Secret in a certain base and splits it into K blocks, i.e., b_0, b_1, ... b_{K-1} with each block consisting of n values. The sender then sends these K blocks to the cloud. Before the block is uploaded to the cloud for storage, the procedure is repeated for a greater number of blocks, during which time one of the files from the file lists is substituted for each value included inside the block. Each block has its own unique file list that has been allocated to it. In addition, copies of these are stored in the cloud. The receiver, who has access to the cloud, looks through the files to determine how to get the data and then produces the Secret by exchanging the information in the files with values.

Despite its suitability for protecting Secrets across multiple clouds from unauthorized access, this mechanism has a limitation, i.e., file selection and storage follow a predictable serial order. This vulnerability exposes the possibility of intruders cracking the Secret. Attackers could exploit this sequential pattern to deduce the concealed message through multiple attempts. Recent work examines this approach's security strength against brute force attacks [1]. The analysis concludes that despite an attacker's attempts to retrieve the Secret value via brute force attacks, the computational effort required for unauthorized access is smaller than exponential computations. In response, the paper seeks to develop a mechanism to make brute-force attacks computationally exponential, proposing "Product Cipher-Based Distributed Steganography".

Selecting a product cipher as the underlying mechanism introduces complexity and security to address concerns in cloud-based communication. The product cipher, a cryptographic construction combining multiple substitution and transposition methods, offers a robust and multi-layered defense against potential attacks. By leveraging this approach, the system aims to counter various intrusion forms, including brute-force attacks - a significant concern in cloud security.

Although the paper doesn't explicitly compare the chosen Product-Cipher-Based Distributed Steganography with other techniques, it primarily focuses on presenting and validating this novel approach. It highlights the unique contributions, such as maintaining unchanged cover media, leveraging multi-cloud storage, and addressing traditional steganography limitations. Further research could delve into comparative studies, evaluating the strengths of the product cipher-based method in contrast to other steganography approaches. Such

comparisons would provide insights into the specific advantages and trade-offs of this technique, situating it within the broader landscape of secure cloud communication steganography methods.

The proposed steganography method, termed "Product Cipher-Based Distributed Steganography" (PCDS), introduces several distinctive features that set it apart from existing methods. The unique aspects of PCDS in contrast to traditional and distributed steganography approaches are:

1. **Unchanged Cover Media as a Reference:**
   - Traditional Steganography: In conventional methods, data is hidden in different cover media, introducing the risk of detection as alterations to cover media may leave traces.
   - Distributed Steganography: Existing distributed methods fragment the Secret message across various cover media, enhancing security but still susceptible to suspicion or loss of the entire Secret in case of modifications.
   - PCDS Approach: PCDS innovatively employs unchanged cover media as a reference for fragmented data. This departure from tradition minimizes the risk associated with modifying cover media, reducing the potential for detection by attackers.
2. **Multi-Cloud Storage Security:**
   - Traditional Steganography: The shift towards cloud usage has prompted individuals to hide private information in images, but concerns remain about the security of sensitive data in the cloud.
   - Distributed Steganography: While distributed methods enhance security, they may still face challenges such as potential detection and loss of the entire Secret.
   - PCDS Approach: PCDS utilizes multi-cloud storage to store the hidden message securely. This safeguards against unauthorized access and introduces mathematical complexity, making it computationally infeasible for attackers to retrieve the concealed data.
3. **Computational Infeasibility for Attackers:**
   - Traditional and Distributed Steganography: The comment alludes to concerns raised in recent research [1] regarding the efficiency of retrieving Secret data from multi-cloud storage. Existing methods may face vulnerabilities to computational attacks.
   - PCDS Approach: PCDS addresses these concerns by introducing a Product Cipher-Based Distributed Steganography scheme. The computational complexity involved in determining the appropriate sequence of Secret distribution and file numbering makes brute-force attacks computationally exponential, ensuring a significantly higher level of security.

Consider a practical example of the PCDS methodology in operation. Consider that Alice wishes to transmit a confidential message to Bob via cloud storage in a secure manner. By employing conventional steganography, she would pose a risk of detection if the image is altered while concealing the message within it. Message fragmentation across multiple images would result from distributed steganography; however, suspicion or alterations could still result in the disclosure of the complete Secret. Using PCDS, Alice can now securely store the fragmented message in a multi-cloud environment while referencing an unchanged image. Thus, in the event of a single cloud compromise, the assailant will not possess the entirety of the information, and the cover media remains unaltered, providing an additional level of security.

The PCDS method not only overcomes traditional and distributed steganography limitations by leveraging unchanged cover media and multi-cloud storage but also establishes a higher level of security through its unique product cipher-based approach. The following sections of the paper will delve deeper into the specifics of PCDS, highlighting its advantages and demonstrating how it effectively addresses the identified shortcomings in existing steganography methods.

2. EXISTING WORK

Steganography, a pivotal component of information security, has garnered substantial interest for its ability to exchange information through various media channels clandestinely. Simmons [19] illuminated the intricacies of maintaining covert communication to evade potential adversaries [37]. This analogy likens the process to a puzzle where Alice and Bob, striving to escape captivity, must communicate discreetly without alerting their captor. The mechanism employed to achieve this covert communication is termed a covert channel, acting as a conduit to shield messages from prying eyes.

Classical steganography is characterized by two essential phases: embedding and extraction. During embedding, a confidential message is subtly integrated into a regular medium, such as text, images, audio, video, or network protocols, employing a shared key. This amalgamation of concealed data and regular mediums engenders a hidden version known as the "stego medium." Subsequently, extraction involves retrieving the concealed message from the stego medium using both the medium itself and the shared key. The primary objective is to remain covert; should an intruder uncover and extract the hidden message, the entire communication becomes vulnerable. Hence, maintaining secrecy is paramount [20, 21].

Nevertheless, specific circumstances can undermine the effectiveness of steganography. If adversaries become aware of the usage of hidden messages through the stego medium, suspicions may arise, especially when messages traverse insecure channels. Additionally, adversarial investigation into message content, often achieved through steganalysis, can compromise concealed communication. Such discovery or extraction of Secret messages contradicts the core tenets of steganography. In severe cases, attackers might manipulate or disable the hidden message, eroding the credibility of the entire discourse.

To counteract covert communication, various steganalysis tools have emerged, particularly targeting images, audio files, and network communications. These tools leverage sophisticated statistical tests like higher-order statistics, Markov random fields, and wavelet statistics to uncover hidden messages [34-36]. Network communications are not exempt, with methods like second-order statistical analysis uncovering concealed channels. Nevertheless, adversaries could still exploit knowledge of communication and delve into message contents, posing vulnerabilities.

To surmount the limitations intrinsic to classical steganography, the concept of distributed steganography [22, 27] has evolved into a sophisticated paradigm. This approach

entails fragmenting a Secret message and dispersing it across disparate hidden media, significantly heightening the challenge of detecting the complete Secret. This strategy proves especially valuable in scenarios where multiple independent senders intend to communicate with a solitary recipient. The advent of cloud technologies has facilitated the concealment of sensitive data within an assemblage of cloud-stored images. Specialized algorithms adeptly disseminate these Secret fragments across the images, mitigating the risk of detection through dispersion among varied media formats.

A pioneering model introduced by Liao et al. [33] and colleagues envisions a distributed steganography scenario where numerous senders interact with a single recipient. Each sender exclusively possesses their covert message, and the amalgamation of these concealed messages can only be deciphered by the intended recipient. This concept closely resembles the concept of Secret sharing. The mechanics involve generating shared keys, distributing them, and subsequently reconstructing the Secret message through these distributed shares. The security and efficiency of such schemes are finely calibrated through diverse strategies, often aligning with cryptographic protocols to ensure robust protection.

However, it's imperative to recognize that distributed steganography confronts its own challenges. While distributing Secrets across varied media heightens security, manipulating these media to hide Secrets could inadvertently attract attention [27]. This vulnerability underscores the significance of blind steganalysis, which aims to unveil new embedding techniques without prior knowledge. By achieving secrecy without altering the original media, a steganographic method can remain concealed even from advanced detection techniques.

Classical steganography and its distributed counterpart offer distinct avenues for Secret communication. Classical steganography conceals messages within individual media, while distributed steganography fragments and disperses Secrets across multiple media for heightened security. The true challenge lies in achieving communication that evades detection while preserving message integrity, particularly in the face of advanced detection methods. These complexities foster innovation, potentially yielding novel steganographic approaches capable of circumventing existing detection mechanisms.

Distributed data hiding in the multi-cloud storage environment presents a novel approach to secure data distribution. Steganography involves concealing confidential information within cover media in a manner that is challenging for adversaries to discern. The proposed method ensures that cover media remain unaltered and act as indicators of fragmented data, bolstering distribution security.

Key Components:

1. **Cover Media Selection and Upload:** The sender selects cover media for concealing the Secret message. These files, resembling typical cloud storage items, are uploaded without modification.
2. **Key Sharing:** Communicators share a key encompassing the number and order of clouds, login credentials, file lists, and an encoding base value. This key transmission ensures authorized access to the hidden data.
3. **Secret Encoding and Distribution:** The sender transcodes the Secret into a specific base and splits it into blocks. Each block value is replaced with a file from the corresponding file list. The modified blocks are distributed to respective clouds.

Based on the above key components, recent work designed "Distributed Data Hiding in the Multi-Cloud Storage Environment." the focus is on hiding a Secret message within a multi-cloud storage environment without directly modifying the original data (cover media) [37]. The concept is built upon two main technical contributions:

1. **Non-modification of Cover Media:** The cover media, which could be any digital content such as images, videos, or files, is not altered in any way. Instead, it acts as a pointer or reference to fragmented data where the Secret message is hidden.
2. **Secret Message Storage:** The Secret message is hidden within the multi-cloud storage environment. The text suggests that the method used to hide the message is complex enough to make it difficult for attackers to detect and extract the Secret message. The concept claims to provide security against certain attacks, particularly brute force attacks.

Despite its suitability for protecting secrets across multiple clouds from unauthorized access, this mechanism has a limitation—file selection and storage follow a predictable serial order. This vulnerability exposes the possibility of intruders cracking the Secret. Attackers could exploit this sequential pattern to deduce the concealed message through multiple attempts. Recent work examines this approach's security strength against brute force attacks [1]. The analysis concludes that despite an attacker's attempts to retrieve the Secret value via brute force attacks, the computational effort required for unauthorized access is smaller than exponential computations. In distributed steganography, if an attacker identifies and modifies one of the cover media elements, the entire Secret may be lost. For instance, if a set of images containing fragments of the message is modified or deleted, Bob may be unable to reconstruct the original message. Additionally, traditional steganography using modified cover media might be vulnerable to steganalysis techniques that exploit alterations in the carrier file.

To further enhance security, the paper proposes an innovative solution - leveraging "Product Cipher-Based Distributed Steganography". Recognizing the imperfections in the existing approach, the proposed mechanism seeks to transform the landscape of security measures against brute-force attacks in a more effective way. By incorporating a product cipher as the underlying mechanism, the system introduces complexity and security measures to address concerns prevalent in cloud-based communication. The product cipher, a cryptographic construction that artfully combines multiple substitution and transposition methods, offers a robust and multi-layered defense against potential attacks. This strategic approach not only thwarts brute-force attacks but also serves as a stalwart guardian against various intrusion forms, aligning with the paramount concerns of cloud security.

In advancing the field of secure communication within cloud environments, the paper not only acknowledges the vulnerabilities posed by predictable patterns but also takes a proactive stance to mitigate them. Through the introduction of a product-cipher-based approach, the system creates an intricate tapestry of protection, ensuring that unauthorized access and potential breaches remain distant possibilities. This innovative approach speaks to the broader efforts of the security community to craft dynamic and resilient safeguards that stand up to the ever-evolving landscape of digital threats.

Consider a practical example of the PCDS approach in action. Suppose Alice wants to securely transmit a confidential message to Bob using cloud storage. In traditional steganography, she might embed the message in an image, risking detection if the image is altered. Distributed steganography would fragment the message across multiple images, but suspicion or modifications could still lead to the loss of the entire Secret. Now, with PCDS, Alice can use an unchanged image as a reference and securely store the fragmented message in a multi-cloud environment. This means even if one cloud is compromised, the attacker won't have the complete information, and the unchanged cover media serves as an added layer of security.

3. PRODUCT CIPHER-BASED DISTRIBUTED STEGANOGRAPHY

In today's digital landscape, the need for secure and covert communication has become increasingly essential. However, ensuring the confidentiality of sensitive information in a distributed manner presents significant challenges. Traditional encryption techniques provide a level of security but are vulnerable to attacks due to their distinguishable patterns and metadata. This necessitates the exploration of novel methods that go beyond conventional encryption. Covert communication involves the exchange of information without arousing suspicion from unintended observers. Steganography, a subset of covert communication, focuses on hiding Secret data within innocuous cover media, such as images or audio files. The challenge lies in developing techniques that not only embed the data effectively but also ensure that the embedded data remains undetectable.

The prevalence of cloud computing has introduced new possibilities for distributed data storage and retrieval. Multi-cloud environments offer benefits such as increased availability and fault tolerance. However, these advantages are accompanied by security concerns, including potential breaches and unauthorized access. Ensuring the confidentiality of data stored across multiple clouds while achieving covert communication further complicates the problem. Existing methods for distributed covert communication often rely on covert channels, which exploit the communication paths not originally intended for data exchange. While effective to a certain extent, these methods have limitations such as susceptibility to detection and difficulty in dynamically adapting to changes in the cloud environment. address these challenges, we propose the Product Cipher-Based Distributed Steganography (PCDS) technique. PCDS aims to securely embed and retrieve data in a multi-cloud environment using steganography, providing both covert communication and resistance against unauthorized access. The approach leverages the concept of product ciphers, combining substitution and permutation operations to achieve secure data hiding. The primary objectives of our research are as follows:

Secure Data Hiding: Develop a technique to dynamically embed Secret data into files distributed across multiple clouds while maintaining the cover media's authenticity.

Covert Communication: Ensure that the embedded data remains undetectable, achieving covert communication between communicating entities.

Resistance to Attacks: Design the PCDS technique to withstand attacks such as statistical analysis and pattern recognition, enhancing the security of the covert communication.

Dynamic Adaptability: Create an approach that adapts to changes in the cloud environment, providing a reliable and secure covert communication channel even in dynamic scenarios.

The proposed Product Cipher-Based Distributed Steganography (PCDS) is an extension of the existing covert channel-based distributed data hiding mechanism [27], which hides the Secret dynamically in the multi-cloud environment in a distributed manner. PCDS is transparent to Secret communication between the communicating entities. PCDS uses multi-clouds to store the files which are derived from the corresponding Secret, and further files are uploaded without any modification.

PCDS is a steganographic approach where files are used as covert media to carry the Secret information that is to be securely shared with the destination. To achieve the goal, communication entities agree on the key. The key agreement procedure is out of the scope of our work. The key consists of a set of information, i.e., a Cloud list and their credentials to access them, a Base value, a session key for permutation, and the number of files indexed in different lists for substitution, Eq. (1) shows the key and its information.

$$Key = \{\, Cloud's = C_0C_1..C_n, Base\ value = B_i, where\ i = 2,4,8.., Session\ Key = K_s, Lists = L_0\, L_1L_2L_3 \ldots..L_n, \text{and } Files\ in\ n^{th}\ list = L_n^0, L_n^1, L_n^2 \ldots \ldots L_n^B, \} \quad (1)$$

where,

1. $C_0C_1 \ldots C_n$ are the number of clouds where Secret need to be stored
2. B_i is the base value, to which secrete value is to be converted
3. Session key (K_s), which is used for applying the permutation
4. $L_0\, L_1L_2L_3 \ldots..L_n$ are the lists, where each list consists of B files

$L_n^0, L_n^1, L_n^2 \ldots \ldots L_n^B$ are the number of files in the $n^{th}list$. These files are in the different format.

Figure 1 shows the overview of the proposed Product Cipher-Based Distributed Steganography (PCDS) approach, where the sender converts the Secret into files and dynamically embeds the files into multi-cloud.

In Figure 1, we present a comprehensive overview of the Product Cipher-Based Distributed Steganography (PCDS) approach, illustrating the process of securely storing and retrieving Secret information within a multi-cloud environment. The diagram is designed to provide a visual representation of the key steps involved in the PCDS technique. To facilitate covert communication, the sender initiates the process by converting the Secret information into a series of files. This conversion involves employing a combination of substitution and permutation techniques, transforming the original Secret into a format suitable for embedding.

Once the files are converted, the sender employs dynamic embedding strategies. These strategies involve utilizing both substitution and permutation operations to ensure that the converted files are embedded securely and discreetly across multiple cloud platforms. Upon successful embedding, the receiver, possessing authorized access to the multi-cloud infrastructure, undertakes the process of retrieving the embedded files. The retrieved files are then subjected to the reverse of the conversion process, where permutation and

substitution operations are applied in the opposite order to reconstruct the original Secret information.

The working procedure of PCDS, i.e., securely stores the Secret to multi-cloud and retrieves the Secret from multi-cloud, is explained as follows On the other hand, the receiver with access to the cloud retrieves the files from the cloud and converts them back to Secret by applying substitution and permutation techniques in reverse order.

The proposed PCDS aims to store and retrieve Secrets in/from multi-cloud securely. To achieve the goal, PCDS split into two phases, i.e., 1). Secure Secret storage in the multi-cloud environment, and 2). Secure Secret retrieval from the multi-cloud environment. Consider the two communicating entities, where one entity wants to store the Secret securely, and another wants to retrieve it with the help of the PCDS algorithm securely. To achieve the goal, both entities agree on the key. The key elements are given in Eq. (1). The First phase of the PCDS, i.e., Secure Secret storage in the multi-cloud environment, is explained as follows.

A. Secure Secret storage in the multi-cloud environment

Secure Secret storage in the multi-cloud environment aims to store the Secret in multi-cloud securely by following the steps.

1. Base Conversion: - The Secret is converted in the base value B.
2. Permutation choice 1: The Secret represented in base B undergoes the permutation.
3. Substitution: Permuted output of the Secret represented in base B is substituted with the files available in the different lists. Thus, each Secret value is represented by different files.
4. Permutation choice 2: The Secret represented by various files undergoes the permutation.
5. Allocation: Permuted output is allocated to different clouds.

This initial step involves the user providing the Secret key 'S' that needs to be securely stored. Additionally, the user specifies the number of clouds involved, which is a critical parameter for distributing the Secret information effectively.

Consider communicating entity-1, who wants to securely store the Secret *S*=1111101101000001 in a multi-cloud environment, with a pre-agreed key between communicating entities is;

$$Key = \{\, Cloud's\ = C_0, C_1, C_2, C_3. Base\ value\ (B_i) = B_2\ Session\ Key = (3\ 1\ 2\ 5\ 4\ ), Lists = \ L_0\, L_1 L_2 L_3 \ldots\ldots L_n, \text{ and } Files\ in\ nth\ list = \ L_n^0, L_n^1, L_n^2 \ldots\ldots L_n^B\ , \}.$$

Entity-1 has the Secret value *S*=1111101101000001 and wants to securely store into four clouds, say $C = C_0, C_1, C_2, C_3.$ with base value *B*=2 and $Session\ Key(K_s) = (3\ 1\ 2\ 5\ 4)$ for permutation and a list of files for substitution, which are shown in Table 1. To store the Secret in the cloud securely, entity-1 applies algorithm-1. The working of algorithm-1 is composed of the following steps.

- ***Base conversion***

The Secret key 'S' undergoes a conversion process utilizing a specified base value 'B.' This conversion is a fundamental aspect of the key agreement between communicating entities, adding an extra layer of security to the process

Initially, algorithm 1 sets the base value (*B*) as per the agreed pre-shared key information, here, base value *B*=2 and applies base conversion to the Secret value. The output of the base conversion is placed into an array list $L_1[N]$ as shown in Table 1.

Table 1. Array $L_1[N]$

Index	$L_1[N]$
0	1
1	1
2	1
3	1
4	1
5	0
6	1
7	1
8	0
9	1
10	0
11	0
12	0
13	0
14	0
15	1

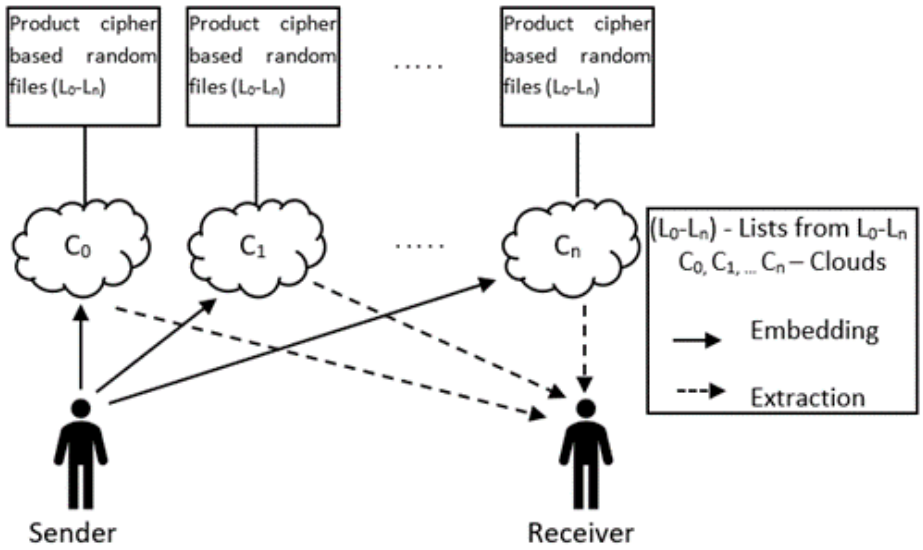

Figure 1. Product cipher based Secret storage in multi-cloud

➢ ***Permuted choice-1***

Permutation choice-1 is applied to the converted Secret key using a session key $'K_s'$ .It's crucial that the session key's length surpasses the number of clouds involved, enhancing the resistance against potential cryptographic attacks.

Algorithm-1 applies permuted choice-1 on an array $L_1[N]$. It selects the session key $'K_s'$ from the pre-shared key. It ensures that the session key length is greater than the number of clouds. If not, increase the key length size. The session key value in the pre-shared key is, i.e., K_s=31254. To apply permuted choice 1, the values of an array $L_1[N]$ converted into the table, as shown in Table 2. If the last row is left with empty cells, fill it with agreed bogus values, i.e., in the example filled with $'0'$. For permutation, read the column-wise as per increment order of key $'K_s'$.

Table 2. Permuted choice –1

3	**1**	**2**	**5**	**4**
1	1	1	1	1
0	1	1	0	1
0	0	0	0	1
1	**0**	**0**	**0**	**0**

The permuted output is $S' = [110\mathbf{0}110\mathbf{0}10011\mathbf{0}1000]$.

➢ ***Substitution***

The permuted output is divided into groups, and substitution is applied to each group based on corresponding lists. These lists play a key role in determining which files represent each value, adding a layer of complexity to the encryption process. The permuted output is divided into groups, each assigned to the list. The list information is available in the pre-shared key, and here, it is shown in Table 3. Eq. (2) is used to form the groups from the permuted output S'. Then each group's values are substituted with lists; each list index values are substituted with files.

$$L_n{}^{S'[nC]} \leq L_n[m < L_n{}^{S'[(n+1)C]}] \quad (2)$$

where,

$n = 0,1,2, \ldots . Ceil\left(\frac{S'}{C} - 1\right)$,
$C = No. of\ Clouds$,
$S' = Length\ of$ permuted output of $Secret\ key$
$L_n = n^{th}\ list$
$m = (n+1)C$
$L_n[m] = nth\ group, which\ consists\ of\ m\ values$
$n^{th} group: L_n[m] =$
$\{L_n^{S'[nC]}, L_n^{S'[nC+2]}, L_n^{S'[nC+2]} \ldots \ldots L_n^{S'[((n+1)C)-1]}\}$

Here, $C = 4, S' = 20$, and if we apply Eq. (2) then it divides the Secret into groups.

$$S'' = \{L_0[4], L_1[4], L_2[4], L_3[4], L_4[4]\}$$

If we apply the corresponding values of each group then, we get:

$$S'' = \{L_0^1, L_0^1, L_0^0, \boldsymbol{L_0^0}, L_1^1, L_1^1, L_1^0, \boldsymbol{L_1^0}, L_2^1, L_2^0, L_2^0, L_2^1, L_3^1, L_3^1, L_3^1, \boldsymbol{L_3^0}, L_4^1, L_4^0, L_4^0, \boldsymbol{L_4^0}\}$$

As per Eq. (2), four lists are created. Those lists names as $L_0[\,], L_1[\,], L_2[\,], L_3[\,]$. Each list contains a group of four values, i.e.,

$$L_0[4] = \{L_0^1, L_0^1, L_0^0, \boldsymbol{L_0^0}\}$$
$$L_1[4] = \{L_1^1, L_1^1, L_1^0, \boldsymbol{L_1^0}\}$$
$$L_2[4] = \{L_2^1, L_2^0, L_2^0, L_2^1\}$$
$$L_3[4] = \{L_3^1, L_3^1, L_3^1, \boldsymbol{L_3^0}\}$$
$$L_0[4] = \{L_0^1, L_0^0, L_0^0, \boldsymbol{L_0^0}\}$$

$L_4[4]$ is converted as $L_0[4]$ as the pre-shared key consist of 4 lists. Further, these array values are substituted with files from the corresponding list based on the index values. The files substitution is shown as follows.

$$L_0[4] = \{Thesis.docs, Thesis.docs, art.docs, art.docs\}$$
$$L_1[4] = \{sheet1.xlsx, sheet1.xlsx, report.xlsx, report.xlsx\}$$
$$L_2[4] = \{file.pptx, document.pptx, document.pptx, file.pptx\}$$
$$L_3[4] = \{linear.pdf, linear.pdf, linear.pdf, publication.pdf\}$$
$$L_0[4] = \{Thesis.docs, art.docs, art.docs, art.docs\}$$

Later substituted file are placed into an array $L_2[N]$ as:

$$L_2[N] = Thesis.docs, Thesis.docs, art.docs, art.docs, sheet1.xlsx, sheet1.xlsx, report.xlsx, report.xlsx, file.pptx, document.pptx, document.pptx, file.pptx, linear.pdf, linear.pdf, linear.pdf, publication.pdf, Thesis.docs, art.docs, art.docs, art.docs\}$$

Table 3. Substituted files in array $L_2[N]$ with corresponding indexes

Index	0	1	2	3	4
$L_2[N]$	$Thesis.docx$	$Thesis.docx$	$art.docs$	$art.docs$	$sheet1.xlsx$
Index	5	6	7	8	8
$L_2[N]$	$sheet1.xlsx$	$report.xlsx$	$report.xlsx$	$file.pptx$	$document.pptx$
Index	10	11	12	13	14
$L_2[N]$	$document.pptx$	$file.pptx$	$linear.pdf$	$linear.pdf$	$linear.pdf$
Index	15	16	17	18	19
$L_2[N]$	$publication.pdf$	$Thesis.docs$	$art.docs$	$art.docs$	$art.docs$

Table 4. Array $O[N]$ after permuted choice – 2

3	**1**	**2**	**5**	**4**
$Thesis.docx$	$Thesis.docx$	$art.docs$	$art.docs$	$sheet1.xlsx$
$sheet1.xlsx$	$report.xlsx$	$report.xlsx$	$file.pptx$	$document.pptx$
$document.pptx$	$file.pptx$	$linear.pdf$	$linear.pdf$	$linear.pdf$
$publication.pdf$	$Thesis.docs$	$art.docs$	$art.docs$	$art.docs$

Table 3 is the array $L_2[N]$ representation in the tabular form.

- ***Permuted choice-2***

Another permutation operation is applied to the substituted values, providing an additional level of security to the process. This step contributes to the robustness of the algorithm against various cryptographic vulnerabilities

Further, algorithm-1 Applies permuted choice-2 on an array $L_2[N]$ same as the permuted choice-1

Table 4 filled in a rectangle row-wise from an array $L_2[N]$. For permutation, read the column-wise as per increment order of key $'K'$ the permuted output

$$O[n] = [Thesis.docx, report.xlsx, file.pptx, Thesis.docs, art.docs, report.xlsx, linear.pdf, art.docs, Thesis.docx, sheet1.xlsx, document.pptx, publication.pdf, sheet1.xlsx, document.pptx, linear.pdf, art.docs, art.docs, file.pptx, linear.pdf, art.docs].$$

Compare with Table 5 indexes remain the same, but files are changed, which is shown in Table 5.

- ***Allocation***

The allocated permuted output is distributed across different clouds based on pre-shared key information. This strategic allocation is designed to ensure that the Secret information is dispersed securely across multiple clouds, preventing a single point of compromise. The final step of algorithm 1 is to allocate the $O[N]$ values into the four clouds C_0, C_1, C_2, C_3, as per the pre-shared key cloud information. Eq. (3) is used to allocate the $O[N]$ values into the different clouds.

$$Position\ allocation:\ O[N] = C_{i,j} \tag{3}$$

where,

$j = N\ mod\ C,$

$i = \frac{N-j}{C},$

$N = 0,1,2 \ldots length(S).$

If N equal to 0 indexed value, then after substituting in Eq. (3) the position of a 0 indexed value, i.e., $Thesis.docx$ is allocated in $C_{0,0}$ of $n*n$ matrix. Similarly, all the indexed file positions are computed and allocated their respective positions. Here is the example of 0 indexed valued file and 19 indexed file:

$$O[0] = C_{0,0} \rightarrow j = 0\ mod\ 4 = 0, i = \frac{0-0}{4} = 0$$
$$O[19] = C_{4,3} \rightarrow j = 19\ mod\ 4 = 3, i = \frac{19-3}{4} = 4$$

Table 5. Permuted output array $O[N]$ with modified files from array $L_2[N]$

$Index$	0	1	2	3	4
$O[N]$	$Thesis.docx$	$report.xlsx$	$file.pptx$	$Thesis.docs$	$art.docs$
$Index$	5	6	7	8	9
$O[N]$	$report.xlsx$	$linear.pdf$	$art.docs$	$Thesis.docx$	$sheet1.xlsx$
$Index$	10	11	12	13	14
$O[N]$	$document.pptx$	$publication.pdf$	$sheet1.xlsx$	$document.pptx$	$linear.pdf$
$Index$	15	16	17	18	19
$O[N]$	$art.docs$	$art.docs$	$file.pptx$	$linear.pdf$	$art.docs$

Table 6. $n*n$ matrix for cloud allocation

$C_{i,j}$	j				
		0	1	2	3
i	0	$Thesis.docx$	$report.xlsx$	$file.pptx$	$Thesis.docs$
	1	$art.docs$	$report.xlsx$	$linear.pdf$	$art.docs$
	2	$Thesis.docx$	$sheet1.xlsx$	$document.pptx$	$publication.pdf$
	3	$sheet1.xlsx$	$document.pptx$	$linear.pdf$	$art.docs$
	4	$art.docs$	$file.pptx$	$linear.pdf$	$art.docs$

Table 7. Column-wise files allocation into multi-clouds

C_0	C_1	C_2	C_3
$Thesis.docx$	$report.xlsx$	$file.pptx$	$Thesis.docs$
$art.docs$	$report.xlsx$	$linear.pdf$	$art.docs$
$Thesis.docx$	$sheet1.xlsx$	$document.pptx$	$publication.pdf$
$sheet1.xlsx$	$document.pptx$	$linear.pdf$	$art.docs$
$art.docs$	$file.pptx$	$linear.pdf$	$art.docs$

Table 8. Files reallocation in $n*n$ matrix

$C_{i,j}$	j				
		0	1	2	3
i	0	$Thesis.docx$	$report.xlsx$	$file.pptx$	$Thesis.docs$
	1	$art.docs$	$report.xlsx$	$linear.pdf$	$art.docs$
	2	$Thesis.docx$	$sheet1.xlsx$	$document.pptx$	$publication.pdf$
	3	$sheet1.xlsx$	$document.pptx$	$linear.pdf$	$art.docs$
	4	$art.docs$	$file.pptx$	$linear.pdf$	$art.docs$

Algorithm-1: Multi-Cloud Secret Key Storage

Step 1: Submit the Secret key $'S'$ and set the no. of clouds.
Step 2: Secret key $'S'$ converted through Base value $'B'$.

$$B_2B_4B_9B_{16} \ldots\ldots B_n$$

Step 3: After conversion into $'B'$, 'S' values placed in array $L_1[N]$ and apply permutation.
Step 4: Choose key $'K'$ for permutation, where $'K'$ length size is greater than no. of clouds.

$$Check\ key\ length\ K\ > No.of\ Clouds$$
$$if\ K\ length\ size < number\ of\ clouds$$
$$then\ K\ length\ size++$$

Step 5: Substitute a group of permuted results with corresponding lists. The group formed by using $nC < L_n \leq (n+1)C$,

$$where, n = 0,1,2, \ldots\ldots Ceil\left(\frac{S}{C} - 1\right), C = No.of\ Clouds, S = Length\ of\ Secret\ key$$
$$Group's: L_0[\,]\ L_1[\,]\ L_2[\,]\ L_3 \ldots . L_n[\,]$$
$$Group: L_n[m] = \{L_n^0, L_n^1, L_n^2 \ldots\ldots L_n^{m-1}\}\ where\ m = no.of\ values\ in\ a\ single\ list\ or\ group$$

Step 6: Grouped lists are substituted with corresponding files $f_1f_2f_3 \ldots\ldots f_n$.
Step 7: Files are placed in an array $L_2[N]$ and apply permutation.

$$L_2[N] = [f_1f_2f_3 \ldots\ldots f_n]$$

Step 8: Permuted files, i.e., $O[N] = [f_1f_2f_3 \ldots\ldots f_n]$ are placed in $n * n$ matrix by computing the position.

$$Position\ allocation: O[N] = C_{i,j}$$
$$where\ j = N\ mod\ C\ , i = \frac{N-j}{C}\ , N\ (files\ indexed\ numbers) = 0,1,2, \ldots\ldots length(S)$$

Step 9: Each column of $n * n$ matrix is stored in the selected cloud.

$$C_1C_2C_3C_4 \ldots\ldots C_n$$

Algorithm 1. Multi-Cloud Secret Key Storage Algorithm

Table 6 shows the position of each Secret key value allocation in the *n*n* matrix by indexing with different files. Attackers *are* unable to notice that Secret values are covered with the file. After file position allocation, select the four clouds $C_0C_1C_2C_3$ to store the part of the Secret key. Each column of *n*n* matrix is stored in the selected cloud, and it is shown in Table 7.

The step-by-step working procedure of secure Secret storage in a multi-cloud environment is shown in Algorithm 1 and Figure 2.

B. Secure Secret retrieval from the multi-cloud environment

The algorithm initiates by identifying and selecting the multi-cloud storage where the Secret information is distributed. This step is critical for initiating the retrieval process. Secure Secret retrieval from the multi-cloud environment aims to retrieve the Secret in the multi-cloud securely by following the steps.

1. Extraction: - Retrieve the files from multiple clouds and place them into the array.
2. Inverse Permutation choice-2: The Secret represented in the array undergoes the inverse permutation with the help of the session key.
3. Substitution: The files available in the output of Inverse Permutation choice-2 are substituted with the list and index values.
4. Inverse Permutation choice-1: The outvalue of the substitution phase undergoes the Inverse Permuted Choice 1 with the help of the session key.
5. Decoding: The Secret value retrieved from Inverse Permutation choice-1 is decoded by the agreed base value from the pre-shared key.

- ***Extraction***

Retrieve the files from multiple clouds and organize them in a matrix form. Each file occupies its designated location in the matrix, forming the basis for subsequent decryption steps. The information available in the multi-clouds in a matrix form, with size *n*m*. The user browses the multi-clouds and fetches the stored information from the cloud, and stores it in the array *O*[*N*]. The location in which the fetched information is stored in array *O*[*N*], is computed by Eq. (4).

$$N = i * C + j \quad (4)$$

where,
$j = column\ number$
$i = row\ number$
$C = number\ of\ clouds$

In our example, the extracting entity (entity-2) browses the multi-clouds i.e., $C_0C_1C_2C_3$, where the files are stored in a matrix form of size 5*4, as shown in Table 8. The files are stored in the matrix with index values $(i,j) = (0,0)(0,1)(0,2) \ldots\ldots\ldots\ldots (4,3)$. Then entity-2 pic files one by one from the matrix and computes the index value using the Eq. (4), and then placed them into the array *O*[*N*].

$$O[N] = [\,Thesis.docx, report.xlsx, file.pptx, Thesis.docs, art.docs, report.xlsx, linear.pdf, art.docs, Thesis.docx, sheet1.xlsx, document.pptx, publication.pdf, sheet1.xlsx, document.pptx, linear.pdf, art.docs, art.docs, file.pptx, linear.pdf, art.docs]$$

- ***Inverse permuted choice-2***

Apply the inverse of permutation choice-2 to the retrieved files. This step involves reversing the permutation operation applied during the storage phase, an essential part of the decryption process.

Apply inverse permuted choice-2 on an array *O*[*N*] using K_s=31254. In the process of inverse permutation, array values *O*[*N*] placed in the column-wise as per increment order of key $'K'$ by using Eq. (5)

$$Column\ length = \frac{\text{array } O[N]\ length}{Key\ length} = \frac{S}{K} \quad (5)$$

Table 9. Inverse permuted choice-2

3	1	2	5	4
Thesis.docx	*Thesis.docx*	*art.docs*	*art.docs*	*sheet1.xlsx*
sheet1.xlsx	*report.xlsx*	*report.xlsx*	*file.pptx*	*document.pptx*
document.pptx	*file.pptx*	*linear.pdf*	*linear.pdf*	*linear.pdf*
publication.pdf	*Thesis.docs*	*art.docs*	*art.docs*	*art.docs*

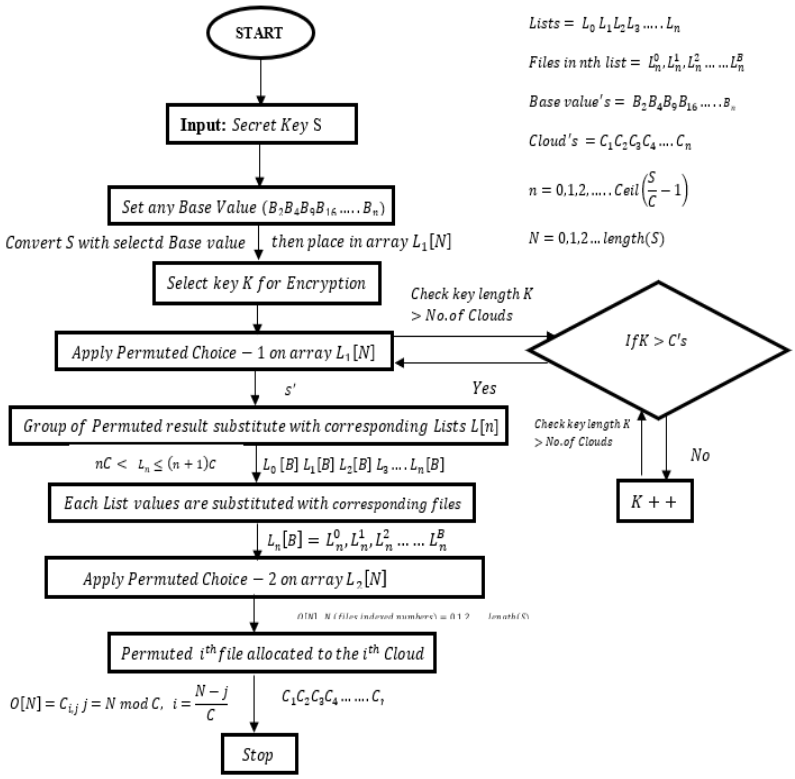

Figure 2. Flowchart of the Secret key storage in multi-clouds

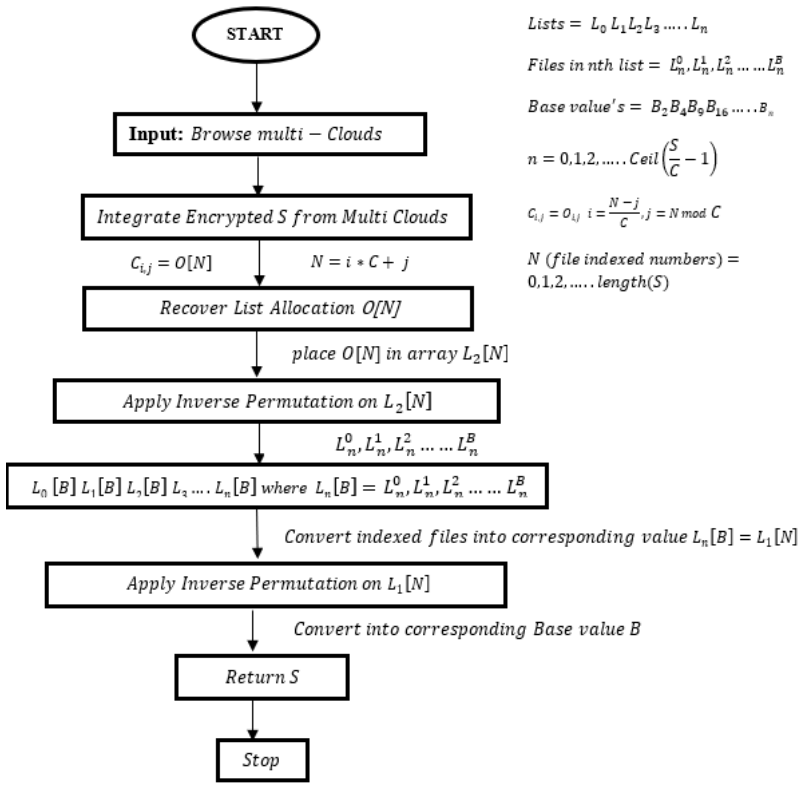

Figure 3. Secret key extraction in multi-clouds

$$Column\ length = \frac{20}{5} = 4$$

In our example, the length of array $O[N]$ is 20 from 16 length of key K=5. Column length is computed by substituting the values of array length, and key length in Eq. (5), and obtained depth of the column is 4.

Insert the first four files of the array $O[N]$ into the column of the key labeled 1, next four files of the array $O[N]$ are inserted into the column of the key labeled 2, and so on, as shown in the Table 9.

The output values of inverse permuted choice-2 are stored in an array $L_2[N]$. The values of $L_2[N]$ are fetched from Table 9 as left to right and top to bottom.

$$L_2[N] = \{Thesis.docs, Thesis.docs, art.docs, art.docs, sheet1.xlsx, sheet1.xlsx, report.xlsx, report.xlsx, file.pptx, document.pptx, document.pptx, file.pptx, linear.pdf, linear.pdf, linear.pdf, publication.pdf, Thesis.docs, art.docs, art.docs, art.docs\}.$$

- ***Substitution***

Substitute the files with list and index values based on pre-shared key information. This step is a reversal of the substitution process applied during the storage phase, restoring the original representation of the Secret information.

The files available in the output of inverse permutation choice-2 are substituted with the list and index values from the pre-shared Secret key. The values of $L_2[N]$ array, i.e., files are substituted by the list numbers and index numbers from Table 1 as follows.

$$L_2[N] = \{L_0^1, L_0^1, L_0^0, \boldsymbol{L_0^0}, L_1^1, L_1^1, L_1^0, \boldsymbol{L_1^0}, L_2^1, L_2^0, L_2^0, L_2^1, L_3^1, L_3^1, L_3^1, \boldsymbol{L_3^0}, L_4^1, L_4^0, L_4^0, \boldsymbol{L_4^0}\}$$

Further, the elements in the $L_2[N]$ the just index values replace array, i.e., remove the lists information and store it in the array $L_1[N]$ as follows.

$L_1[N]$=[11001100100111101000]

- ***Inverse permuted choice-1***

Apply the inverse of permutation choice-1 to further decrypt the retrieved information. This step is crucial for unraveling the encryption layers applied during the storage phase.

Apply inverse permuted choice-1 on an array $L_1[N]$ using session key K_s=31254. In the process of inverse permutation, array values of $L_1[N]$ are placed in the column-wise as per increment order of key $'K'$ by using Eq. (5), as follows; and shown in Table 10.

Table 10. Inverse permuted choice-1

3	**1**	**2**	**5**	**4**
1	1	1	1	1
0	1	1	0	1
0	0	0	0	1
1	**0**	**0**	**0**	**0**

The output of the inverse permuted choice-2 are stored in list S. The values of list S are fetched from Table 10 as left to right and top to bottom, as follows:

S=11111011010000010000

Remove agreed-on bogus values and place the values in row-wise order from the Table 10 to get the Secret key.

S=1111101101000001

- ***Decoding:***

Decode the retrieved information using the agreed base value from the pre-shared key. This final step transforms the information back to its original form, completing the secure retrieval process. The agreed base value decodes the Secret value retrieved from Inverse Permutation choice-1. The agreed base value is B=2. Thus the decoded value is S=1111101101000001.

The step-by-step working procedure of secure Secret extraction in a multi-cloud environment is shown in Algorithm 2 and Figure 3.

Algorithm:2 Multi-Cloud Secret Key Extraction

Step 1: Browse and find selected multi cloud storage.

Step 2: Extract all permuted files from multi clouds and allocate them in their respective locations in $n * n$ matrix

$$location: C_{i,j} = O_{i,j}\ where\ \{i,j\} = (0,0)\ (0,1)\ (0,2) \dots \dots \dots \dots (p,q)$$
$$where\ i = \frac{N-j}{C}, j = N\ mod\ C, N\ (files\ indexed\ numbers) = 0,1,2, \dots . . length(S)$$
$$N = i * C + j$$

Step 3: Select the files row wise from $n*n$ matrix and place into array $L_2[N]$.

Step 4: Apply inverse permutation on $L_2[N]$.

Step 5: Group of files $[f_1 = L_n^0, f_2 = L_n^0, f_3 = L_n^0 \dots . . f_n = L_n^B]$ substitute in a list.

$$L_n[B] = L_n^0, L_n^1, L_n^2 \dots \dots L_n^B$$

Step 6: Arrange list $L_n[B]$ in a row wise by

$$nC < L_n \leq (n+1)C$$
$$where\ n = 0,1,2, \dots . . Ceil\left(\frac{S}{C} - 1\right), C = No.of\ Clouds, S = Length\ of\ Secret\ key$$

Step 7: Indexed files are replaced with corresponding values and placed into array $L_1[N]$ and apply inverse permutation for decryption.

$L_1[N]$=B values

Step 8: B values convert into base value S.

Algorithm 2. Multi-Cloud Secret Key Extraction Algorithm.

4. RESULTS AND DISCUSSION

The paper presents a Product Cipher-Based Distributed Steganography scheme designed to dynamically hide Secrets in a multi-cloud environment. This undetectable Secret distribution system relies on steganography, a method of concealing information to make it hard to find. In contrast to related steganographic work, our approach to Secret extraction in a multi-cloud storage environment does not depend on

modifying files to hide the presence of a covert channel between communicating entities. The scheme facilitates efficient Secret distribution in a multi-cloud environment without modifying cover files, ensuring the original files remain unchanged. This minimizes overhead and avoids any suspicious content that could attract adversaries' attention.

Additionally, our method refrains from using special characters, such as space, ASCII code letter A0, character coloring, or text justification, to conceal information. As a result, inspecting the file content reveals no questionable elements, preventing the file content from drawing the adversary's attention. This enhances the scheme's efficiency by eliminating the need for sophisticated encoding or decoding methods, reducing computational cost, and ensuring faster and more effective Secret extraction.

PCDS is a secure method for storing and retrieving Secrets in a multi cloud environment. Its security can be assessed using the following criteria:

1. Confidentiality: PCDS seeks to protect the secrecy of the stored Secret by converting it to files and embedding them in multiple clouds. The use of permutation and substitution techniques incorporate the added layer of security. The secrecy of cloud list, base value, session key, and file lists is essential to provide the confidentiality.
2. Covert Channel Detection: The maintenance of an undetectable covert channel between communicating entities is prioritized by PCDS. The technique decreases the risk of an attacker discovering the presence of a covert channel by using existing cover files without modification and avoiding suspicious material. This improves the overall security of the system.
3. Computational Complexity: PCDS's security is based on the computational complexity required for an attacker to extract the Secret. Permutations, substitutions, and several files in distinct lists are used in the method, making it computationally difficult for an adversary to establish the precise distribution and numbering of Secret information. Even if all cloud accounts are available, recovering the Secret is computationally unfeasible due to its complexity.

Lemma: By maintaining the secrecy of crucial elements and employing strong encryption, the PCDS scheme ensures it is computationally impossible for an attacker to recover the original Secret without the encryption key.

Proof: The PCDS scheme achieves confidentiality through two key features:

1. Secrecy of Key Elements: The PCDS scheme protects the cloud list, base value, session key, and file lists from unauthorized access. These crucial elements remain hidden to prevent attackers from learning about encryption or file delivery across clouds. Without these confidential key pieces, an adversary cannot decrypt the Secret.
2. Robust Encryption Techniques: The PCDS scheme uses product cipher-based encryption to store the original Secret in encrypted files distributed across multiple clouds. Even if an adversary gains access to encrypted files, these encryption methods render the Secret information unreadable. The encryption key is kept Secret from attackers. The complexity of encryption makes it computationally impossible to interpret encrypted data and retrieve the original Secret without the correct encryption key.

The PCDS approach safeguards the Secret by keeping essential aspects hidden and employing strong encryption. This strategy, which combines maintaining key secrecy and using robust encryption, creates significant computational obstacles for attackers, thereby protecting multi-cloud Secret information

Lemma: The PCDS mechanism maintains an undetectable covert channel by using existing cover files without modification and avoiding suspicious content.

Proof: The PCDS scheme ensures covert channel detection through two techniques:

1. **Utilization of Existing Cover Files:** The scheme leverages existing cover files in the multi-cloud environment without modifying them. By utilizing these legitimate files for a genuine purpose, the scheme avoids arousing suspicion and effectively conceals the presence of a Secret channel.
2. **Avoidance of Suspicious Content:** The PCDS scheme refrains from using special characters, space, ASCII code letter A0, character coloring, or text justification to conceal information. This approach eliminates any questionable content that may attract attention during file inspection, enhancing performance, reducing computational overhead, and ensuring efficient Secret extraction.

By using unmodified cover files and avoiding suspicious content, the PCDS scheme ensures an undetectable covert channel. Adversaries analyzing the file content will find no indications of a Secret channel, making it highly improbable for them to detect the hidden communication.

Lemma: The PCDS scheme's computational complexity makes it computationally infeasible for an attacker to retrieve the Secret, even with full access to all cloud accounts.

Proof: The PCDS scheme employs permutations, substitutions, and multiple files in different lists to distribute the Secret across the multi-cloud environment. The computational cost required for an attacker to discover the correct Secret distribution and numbering is greatly increased by these elements. Using permutations and substitutions, the scheme generates a large number of possible combinations. This makes it extremely difficult for an attacker to predict the correct Secret distribution without the key. In addition, the Secret's distribution across numerous files and distinct lists increases the computational complexity. The attacker must accurately identify the sequence of Secret distribution and the numbering of files within each list. This involves a vast number of possibilities

Considering these factors, the total number of computations required for the attacker to retrieve the Secret value is given by B! * k! * n!, where B represents the number of potential permutations, k represents the number of substitutions, and n represents the number of files within each list.

This computational complexity is exponential and grows rapidly as the number of permutations, substitutions, and files increases. As a result, it becomes computationally infeasible for the attacker to retrieve the Secret, even with full access to all cloud accounts.

The existing approach in distributed data hiding in the multi-cloud storage environment focuses on two technical contributions: the cover media acting as a pointer to fragmented data and the storage of a Secret message in the multi-cloud environment. It claims to make it complicated for

an attacker to detect and extract the Secret message. However, upon analysis of the security strength of this approach, it has been concluded that it is vulnerable to brute force attacks that require computations significantly smaller than the exponential complexity of B!*K!*n!.

In contrast, the proposed PCDS scheme offers a more robust solution for the secure distribution of Secrets in a multi-cloud environment. The PCDS scheme employs permutations, substitutions, and multiple files in distinct lists, which substantially increases the computational complexity required for an attacker to discover the Secret. Due to the large number of possible combinations introduced by permutations and substitutions, it is extremely difficult for an attacker to predict the correct Secret distribution without the key. In addition, dispersing the Secret across multiple files in distinct lists increases the computational difficulty. The total number of computations necessary to retrieve the Secret is given by B!*K!*n!, where B represents potential permutations, k represents substitutions, and n represents the number of files contained within each list. This complexity increases exponentially with the number of permutations, substitutions, and files, making it computationally impossible for an attacker to retrieve the Secret, even if they have access to all cloud accounts.

Unlike the existing technique, which is vulnerable to brute-force attacks with computations less than B!*K!*n!, the PCDS provides significantly higher security assurances. Its computational complexity exponentially increases with the number of permutations, substitutions, and files. This ensures secrecy in the multi-cloud environment, providing a robust defense against brute-force attacks.

Overall, the PCDS ensures strong security by employing steganographic techniques, maintaining an undetectable covert media, and making it challenging for potential attackers to perform computations. It offers a realistic and efficient solution for Secret storage and retrieval in a multi-cloud context.

The existing paper analyzes security through two attack hypotheses: Hypothesis 1 involves adversaries without cloud account access, and Hypothesis 2 involves adversaries with varying levels of cloud account access. The existing analysis concludes that adversaries lacking access remain unaware of Secret communication, and even those with access are impeded by the exponential complexity of permutation-based attacks.

In contrast, our proposed work provides a more encompassing security analysis, delving deeper into the scheme's resilience across diverse attack scenarios. This analysis revolves around three pivotal lemmas: confidentiality, preservation of an undetectable covert channel, and computational complexity. Rigorous proofs support each lemma, showcasing the scheme's robustness.

The confidentiality lemma safeguards crucial elements like cloud lists and session keys, ensuring unauthorized decryption is infeasible. The maintenance of an undetectable covert channel is achieved through the use of existing cover files and the avoidance of suspicious content. Notably, the computational complexity lemma underscores the scheme's resilience against adversaries with full access

Comparing the security strengths of both analyses, the proposed work's analysis excels in breadth, rigor, and a dynamic approach. Unlike the existing analysis, which focuses on limited attack scenarios and vulnerabilities, our analysis explores the scheme's resilience from multiple dimensions. The thorough examination of core lemmas demonstrates the scheme's strength against diverse attacks, reinforcing its security.

Addressing theoretical facets like encryption strength, covert channel integrity, and computational complexity, the proposed work establishes a more compelling basis for secure Secret distribution in a multi-cloud environment.

While maintaining an undetectable Secret channel without modifying existing cover files is a noteworthy aspect of our approach, it's crucial to recognize that this feature results from the underlying research principles forming the basis of our scheme. Our fundamental contribution is the development of a novel Product Cipher-Based Distributed Steganography scheme, specifically designed for secure and dynamic Secret distribution within a multi-cloud environment.

It's essential to emphasize that our security argument goes beyond the concealment of the channel itself. Our extensive security analysis adopts a holistic approach, thoroughly examining key security elements, including confidentiality, preservation of an undetectable covert channel, and computational complexity. Through rigorous proofs, we establish the scheme's resilience against a wide range of potential attack scenarios. This comprehensive analysis reinforces our claim that our proposed scheme provides a robust and secure framework for Secret distribution and retrieval in multi-cloud contexts.

By encompassing these broader security dimensions, we offer a comprehensive and robust justification for the effectiveness of our approach.

5. CONCLUSION

The paper introduces a Product Cipher-Based Distributed Steganography for secure Secret distribution in a multi-cloud environment. This approach conceals the Secret by fragmenting it into smaller parts and placing each fragment within separate cover files, utilizing multi-cloud storage without modifying them. Our strategy, validated through a comprehensive security analysis, demonstrates the computational infeasibility for attackers to decipher the hidden message, even with complete access to all cloud accounts. Striking a balance between efficient Secret distribution and robust security, our method eliminates the need for complex encoding, reducing computational overhead and ensuring unaltered original files, mitigating potential suspicion from adversaries.

REFERENCES

[1] Arif, M.A., Mohammad, A.A.K., Sastry, M.K., Bankapalli, J. (2022). Brute force attack on distributed data hiding in the multi-cloud storage environment more diminutive than the exponential computations. Ingenierie des Systemes d'Information, 27(6): 915-921. https://doi.org/10.18280/isi.270607

[2] Zhang, Y., Geng, H., Su, L., Lu, L. (2022). A blockchain-based efficient data integrity verification scheme in multi-cloud storage. IEEE Access, 10: 105920-105929. https://doi.org/10.1109/ACCESS.2022.3211391

[3] Hassan, J., Shehzad, D., Habib, U., Aftab, M.U., Ahmad, M., Kuleev, R., Mazzara, M. (2022). The rise of cloud computing: Data protection, privacy, and open research

challenges - a systematic literature review (SLR). Computational Intelligence and Neuroscience, 2022: 8303504. https://doi.org/10.1155/2022/8303504

[4] Mohd Satar, S.D., Hussin, M., Hanapi, Z.M., Mohamed, M.A. (2021). Towards virtuous cloud data storage using access policy hiding in ciphertext policy attribute-based encryption. Future Internet, 13(11): 279. https://doi.org/10.3390/fi13110279

[5] Gupta, I., Singh, A.K., Lee, C.N., Buyya, R. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. IEEE Access, 10: 71247-71277. https://doi.org/10.1109/ACCESS.2022.3188110

[6] Gaur, M., Jailia, M. (2022). Cloud computing data security techniques - A survey. In: Kumar, A., Srivastava, S.C., Singh, S.N. (eds) Renewable Energy Towards Smart Grid. Lecture Notes in Electrical Engineering, vol 823. Springer, Singapore. https://doi.org/10.1007/978-981-16-7472-3_5

[7] Gutub, A., Alaseri, K. (2020). Hiding shares of counting-based Secret sharing via Arabic text steganography for personal usage. Arabian Journal for Science and Engineering, 45(4): 2433-2458. https://doi.org/10.1007/s13369-019-04010-6

[8] Abdul, A.M., Mohammad, A.A.K., Venkat Reddy, P., Nuthakki, P., Kancharla, R., Joshi, R., Kannaiya Raja, N. (2022). Enhancing security of mobile cloud computing by trust-and role-based access control. Scientific Programming, 2022: 9995023. https://doi.org/10.1155/2022/9995023

[9] Mewada, S. (2023). Cryptic algorithms: Hiding sensitive information in cloud computing. In Encyclopedia of Data Science and Machine Learning, pp. 781-789. https://doi.org/10.4018/978-1-7998-9220-5.ch044

[10] Gutub, A., Al-Ghamdi, M. (2020). Hiding shares by multimedia image steganography for optimized counting-based Secret sharing. Multimedia Tools and Applications, 79(11-12): 7951-7985. https://doi.org/10.1007/s11042-019-08427-x

[11] Ge, X., Yu, J., Hao, R., Lv, H. (2021). Verifiable Keyword search supporting sensitive information hiding for the cloud-based healthcare sharing system. IEEE Transactions on Industrial Informatics, 18(8): 5573-5583. https://doi.org/10.1109/TII.2021.3126611

[12] Aminzade, M. (2018). Confidentiality, integrity and availability–finding a balanced IT framework. Network Security, 2018(5): 9-11. https://doi.org/10.1016/S1353-4858(18)30043-6

[13] Man, Z., Li, J., Di, X., Zhang, R., Li, X., Sun, X. (2023). Research on cloud data encryption algorithm based on bidirectional activation neural network. Information Sciences, 622: 629-651. https://doi.org/10.1016/j.ins.2022.11.089

[14] Alemami, Y., Al-Ghonmein, A.M., Al-Moghrabi, K.G., Mohamed, M.A. (2023). Cloud data security and various cryptographic algorithms. International Journal of Electrical and Computer Engineering, 13(2): 1867-1879. https://doi.org/10.11591/ijece.v13i2.pp1867-1879

[15] Abd-El-Atty, B., ElAffendi, M., El-Latif, A.A.A. (2023). A novel image cryptosystem using Gray code, quantum walks, and Henon map for cloud applications. Complex & Intelligent Systems, 9(1): 609-624. https://doi.org/10.1007/s40747-022-00829-z

[16] Gadde, S., Amutharaj, J., Usha, S. (2023). A security model to protect the isolation of medical data in the cloud using hybrid cryptography. Journal of Information Security and Applications, 73: 103412. https://doi.org/10.1016/j.jisa.2022.103412

[17] Makhdoom, I., Abolhasan, M., Lipman, J. (2022). A comprehensive survey of covert communication techniques, limitations and future challenges. Computers & Security, 120: 102784. https://doi.org/10.1016/j.cose.2022.102784

[18] Denis, R., Madhubala, P. (2021). Hybrid data encryption model integrating multi-objective adaptive genetic algorithm for secure medical data communication over cloud-based healthcare systems. Multimedia Tools and Applications, 80: 21165-21202. https://doi.org/10.1007/s11042-021-10723-4

[19] Simmons, G.J. (1984). The Prisoners' Problem and the Subliminal Channel. In: Chaum, D. (eds) Advances in Cryptology. Springer, Boston, MA. https://doi.org/10.1007/978-1-4684-4730-9_5

[20] Chinnasamy, P., Deepalakshmi, P., Dutta, A.K., You, J., Joshi, G.P. (2021). Ciphertext-policy attribute-based encryption for cloud storage: Toward data privacy and authentication in AI-enabled IoT system. Mathematics, 10(1): 68. https://doi.org/10.3390/math10010068

[21] Ying, Z., Jiang, W., Liu, X., Xu, S., Deng, R.H. (2021). Reliable policy updating under efficient policy hidden fine-grained access control framework for cloud data sharing. IEEE Transactions on Services Computing, 15(6): 3485-3498. https://doi.org/10.1109/TSC.2021.3096177

[22] Moyou Metcheka, L., Ndoundam, R. (2020). Distributed data hiding in multi-cloud storage environment. Journal of Cloud Computing, 9(1): 68. https://doi.org/10.1186/s13677-020-00208-4

[23] Wang, T., Yang, Q., Shen, X., Gadekallu, T.R., Wang, W., Dev, K. (2021). A privacy-enhanced retrieval technology for the cloud-assisted internet of things. IEEE Transactions on Industrial Informatics, 18(7): 4981-4989. https://doi.org/10.1109/TII.2021.3103547

[24] Deepthi, B., Ramani, G., Deepika, R., Shabbeer, M. (2021). Hybrid secure cloud storage data based on improved encryption scheme. In 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, pp. 776-779. https://doi.org/10.1109/ESCI50559.2021.9396842

[25] Wagemann, J., Siemen, S., Seeger, B., Bendix, J. (2021). A user perspective on future cloud-based services for Big Earth data. International Journal of Digital Earth, 14(12): 1758-1774. https://doi.org/10.1080/17538947.2021.1982031

[26] Xie, H., Zhang, Z., Zhang, Q., Wei, S., Hu, C. (2021). HBRSS: Providing high-secure data communication and manipulation in insecure cloud environments. Computer Communications, 174: 1-12. https://doi.org/10.1016/j.comcom.2021.03.018

[27] Marion, N.E., Twede, J. (2020). Cybercrime: An Encyclopedia of Digital Crime. Bloomsbury Publishing USA.

[28] Jiang, S., Ye, D., Huang, J., Shang, Y., Zheng, Z. (2020). SmartSteganogaphy: Light-weight generative audio steganography model for smart embedding application. Journal of Network and Computer Applications, 165: 102689. https://doi.org/10.1016/j.jnca.2020.102689

[29] Sahu, A.K., Swain, G. (2020). Reversible image

steganography using dual-layer LSB matching. Sensing and Imaging, 21: 1-21. https://doi.org/10.1007/s11220-019-0262-y

[30] Pilania, U., Gupta, P. (2020). Analysis and implementation of IWT-SVD scheme for video steganography. In: Sharma, D.K., Balas, V.E., Son, L.H., Sharma, R., Cengiz, K. (eds) Micro-Electronics and Telecommunication Engineering. Lecture Notes in Networks and Systems, vol 106. Springer, Singapore. https://doi.org/10.1007/978-981-15-2329-8_16

[31] Yang, J., Liao, X. (2020). An embedding strategy on fusing multiple image features for data hiding in multiple images. Journal of Visual Communication and Image Representation, 71: 102822. https://doi.org/10.1016/j.jvcir.2020.102822

[32] Liao, X., Yin, J., Chen, M., Qin, Z. (2020). Adaptive payload distribution in multiple images steganography based on image texture features. IEEE Transactions on Dependable and Secure Computing, 19(2): 897-911. https://doi.org/10.1109/TDSC.2020.3004708

[33] Liao, X., Wen, Q.Y., Shi, S. (2011). Distributed steganography. In 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Dalian, China, pp. 153-156. https://doi.org/10.1109/IIHMSP.2011.20

[34] AlKhodaidi, T., Gutub, A. (2020). Trustworthy target key alteration helping counting-based Secret sharing applicability. Arabian Journal for Science and Engineering, 45: 3403-3423. https://doi.org/10.1007/s13369-020-04422-9

[35] Gutub, A., Al-Ghamdi, M. (2019). Image based steganography to facilitate improving counting-based Secret sharing. 3D Research, 10: 1-36. https://doi.org/10.1007/s13319-019-0216-0

[36] Gutub, A., Al-Juaid, N., Khan, E. (2019). Counting-based Secret sharing technique for multimedia applications. Multimedia Tools and Applications, 78: 5591-5619. https://doi.org/10.1007/s11042-017-5293-6

[37] Moyou Metcheka, L., Ndoundam, R. (2020). Distributed data hiding in multi-cloud storage environment. Journal of Cloud Computing, 9(1): 68. https://doi.org/10.1186/s13677-020-00208-4