International Journal of Safety and Security Engineering
Vol. 15, No. 7, July, 2025, pp. 1519-1526
Journal homepage: http://iieta.org/journals/ijsse

Multi-Encoding Distributed Steganography (MEDS): A High-Security and Fault-Tolerant Framework for Secure Data Hiding in Multi-Cloud Environment

Syed Shakeel Hashmi[1], Sireesha Moturi[2], Arshad Ahmad Khan Mohammad[3*], Raj Mohammad Mohd[3], Arif Mohammad Abdul[3], Anusha Marouthu[4]

[1] Department of Electronics and Communication Engineering, Faculty of Science and Technology (IcfaiTech), ICFAI Foundation for Higher Education Hyderabad, Hyderabad 501203, India
[2] Department of CSE, Narasaraopeta Engineering College (Autonomous), Narasaraopeta 522601, India
[3] Department of CSE, School of Technology, GITAM Deemed to be University, Hyderabad 502329, India
[4] Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram 522502, India

Corresponding Author Email: amohamma2@gitam.edu

Copyright: ©2025 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/ijsse.150719

Received: 8 June 2025
Revised: 12 July 2025
Accepted: 20 July 2025
Available online: 31 July 2025

Keywords:
multi-cloud security, distributed data hiding, steganography, private clouds, multi-encoding, dynamic file selection, Fisher-Yates randomization, erasure coding, k-means clustering, security

ABSTRACT

Securing sensitive data in multi-cloud environments is a pressing challenge due to vulnerabilities in traditional and distributed data hiding methods. Classical steganography embeds secrets in modified cover media, leaving detectable artifacts, while distributed approaches, such as the Product Cipher-Based Distributed Steganography (PCDS), are susceptible to brute-force attacks with sub-exponential complexity. This paper proposes the Multi-Encoding Distributed Steganography (MEDS) framework, a novel distributed data hiding paradigm. The process leverages private clouds, multiple base encodings (B_1=2, B_2=4, B_3=8), k-means clustering, authenticated Diffie-Hellman key exchange, and erasure coding. Unlike traditional steganography, MEDS uses unaltered cover files as pointers to fragmented secrets, ensuring imperceptibility and resilience against steganalysis. By employing dynamic file selection and Fisher-Yates shuffles, MEDS achieves an attack complexity of 10^{159}, with 15.2–23.1% faster storage/retrieval times and 13.5% lower memory usage compared to PCDS. Evaluations on an OpenStack cluster demonstrate superior performance, fault tolerance, and scalability compared to PCDS, setting a new benchmark for secure, undetectable data hiding in multi-cloud environments.

1. INTRODUCTION

The proliferation of cloud computing has revolutionized data storage and communication by providing scalable, cost-efficient infrastructure, enabling dynamic resource allocation and global accessibility [1]. However, the adoption of multi-cloud environments [2], where data is distributed across multiple heterogeneous cloud service providers, has significantly amplified cybersecurity threats, such as data breaches, including data theft, and unauthorized surveillance, including espionage [3]. These multi-cloud architectures exacerbate risks due to increased attack surfaces and fragmented security policies, necessitating robust security mechanisms to protect sensitive information. Cryptography ensures data confidentiality by transforming plaintext into cryptographically secure, unreadable ciphertexts using algorithms like AES-256, while steganography and distributed data hiding techniques aim to conceal the very existence of sensitive data, thereby achieving covertness critical for secure multi-cloud communication [4, 5].

Steganography, defined as the practice of embedding secret data within innocuous cover media (e.g., digital images, audio files), achieves covertness by evading detection through steganalysis tools that exploit statistical deviations in media properties [6]. Classical steganography methods, such as least significant bit (LSB) embedding, modify cover media by altering low-order bits, introducing detectable statistical artifacts identifiable with higher accuracy by modern steganalysis algorithms, such as ensemble classifiers or deep convolutional neural networks [7, 8]. Distributed steganography addresses these limitations by fragmenting secret data across multiple cover media, reducing the risk of detection by dispersing statistical anomalies [9]. However, many distributed approaches still require direct modifications to cover files, compromising imperceptibility and limiting scalability in multi-cloud environments due to increased computational overhead and potential synchronization issues.

Existing multi-cloud steganography frameworks leverage unmodified cover files as metadata pointers to encode secret data, achieving perfect imperceptibility by avoiding alterations to file content. However, cryptanalysis has revealed vulnerabilities stemming from predictable file-to-index mappings, enabling adversaries to reconstruct secrets with sub-exponential computational effort (e.g., 4–16 iterations for

encoding bases ($B = 4, 2$) . The Product Cipher-Based Distributed Steganography (PCDS) [10] framework enhances security through cryptographic randomization techniques, such as Fisher-Yates shuffles, to permute file selections. Nevertheless, PCDS remains susceptible to statistical attacks due to predictable file selection patterns, reliance on single-base encoding schemes ($B = 2, 4, 8$), absence of metadata integrity verification, and limited randomization scope, which collectively reduce its resilience against sophisticated adversaries with a 50% success rate in 10^{12} operations for a 16-bit secret. These limitations compromise covertness in high-stakes applications like healthcare.

This paper introduces the Multi-Encoding Distributed Steganography (MEDS) framework, a novel distributed data hiding approach designed to address these vulnerabilities in multi-cloud environments. MEDS leverages private cloud infrastructure, multiple encoding bases ($B_1 = 2$, $B_2 = 4$, $B_3 = 8$), and advanced cryptographic primitives, including secure hash functions and authentication to enhance security and randomization. Unlike classical methods, MEDS uses unaltered cover files as pointers, achieving no detectable artifacts in tests with numerous files. Its dynamic file selection significantly increases entropy compared to PCDS, while multi-base encoding and Fisher-Yates shuffles yield a vastly higher attack complexity, offering a substantial improvement over PCDS. By ensuring perfect imperceptibility, robust security, and fault tolerance, MEDS sets a new benchmark for secure data hiding in multi-cloud environments, with sensitive applications. MEDS introduces four key contributions:

Multiple base encodings: Employs dynamic encoding bases (B_1 =2, B_2 =4, B_3 =8), to transcode secret data, significantly increasing combinatorial complexity and thwarting pattern-based attacks.

Unmodified cover files as pointers: Enhances file selection randomness using k-means clustering of metadata attributes, reducing predictability and ensuring robust imperceptibility [11].

Exponential complexity: Integrates Fisher-Yates shuffles [12], Diffie-Hellman key exchange [13], and erasure coding to achieve a combinatorial attack complexity of 10^{159} i.e., ($B! \cdot K! \cdot n!$) , rendering brute-force and statistical attacks infeasible.

Fault-tolerant storage: Distributes secret fragments across private clouds with erasure coding, ensuring data reliability and availability even under partial cloud failures [14].

Consider an example, Alice transmits a confidential message to Bob by fragmenting it across multiple clouds using MEDS, with unaltered files serving as metadata pointers and randomized multi-base encodings. Even if one or more clouds are compromised, the exponential attack complexity ensures the security of the hidden data.

Steganography's effectiveness is evaluated across four key characteristics: security (resistance to steganalysis), imperceptibility (undetectable modifications to cover media), payload (data embedding capacity), and robustness (resilience to media alterations or transmission errors). Classical LSB-based steganography, which embeds secrets within a single cover medium, compromises imperceptibility and security due to detectable statistical artifacts. Distributed steganography improves security by fragmenting secrets but often retains media modification requirements, risking detection. PCDS advances imperceptibility by using unaltered files as pointers to fragmented secrets but remains vulnerable to brute-force attacks due to predictable file selection patterns and limited encoding diversity. In contrast, MEDS eliminates embedding within file content, utilizing unmodified cover files exclusively as metadata-based pointers to secret fragments distributed across multiple cloud providers. This approach ensures perfect imperceptibility, as no statistical artifacts are introduced, and achieves high security through randomized multi-base encoding and dynamic k-means clustering of file metadata, which exponentially increases the computational complexity of steganalysis and brute-force attacks. MEDS further incorporates erasure coding to provide robust fault tolerance, ensuring data recovery despite partial cloud outages, while maintaining moderate payload capacity suitable for practical applications. By addressing the limitations of prior frameworks, MEDS establishes a resilient, scalable solution for covert data protection in multi-cloud systems, offering enhanced security, imperceptibility, and robustness against both passive and active adversarial threats.

The paper is organized as follows: Section II provides background on steganography and multi-cloud systems, Section III reviews related work, Section IV states the problem and threat model, Section V details the MEDS framework and evaluates performance metrics, Section VI discusses results and implications, Section VII concludes the study, and Section VIII lists references.

2. BACKGROUND KNOWLEDGE

Steganography conceals sensitive data within innocuous cover media, ensuring covert communication, unlike cryptography's focus on confidentiality. Classical steganography, such as the least significant bit (LSB) embedding, modifies a single medium, leaving statistical artifacts detectable by steganalysis. Distributed steganography fragments secrets across multiple media, reducing detection risks but often requiring modifications, exposing data to advanced steganalysis. Multi-cloud environments, where data is distributed across private clouds for scalability and fault tolerance, offer new opportunities for hiding data but amplify security challenges due to fragmented storage.

The Product Cipher-Based Distributed Steganography (PCDS) framework uses unaltered cover files as pointers to be fragmented secrets, mitigating modification vulnerabilities. PCDS hides a secret ($e.g., S = 111110110100101$) by converting it to a base ($B = 2, 4, or\ 8$), shuffling files, and distributing shards across clouds, recoverable with three of four shards. However, its single encoding base and predefined file lists limit combinatorial complexity, enabling brute-force attacks with sub-exponential effort by exploiting metadata patterns. The proposed MEDS framework addresses these limitations by introducing multiple encoding bases (B_1 =2, B_2 =4, B_3 =8), dynamic k-means clustering ($k = 3$), and multistage shuffles, achieving factorial attack complexity ($B! \cdot K! \cdot n!$), ensuring imperceptibility and resilience against steganalysis.

3. RELATED WORK

The field of steganography has seen significant advancements with the rise of multi-cloud environments, driven by the need for security, covert, and scalable data hiding to protect sensitive information from cyber threats like data theft and espionage. This section reviews prior work

relevant to distributed steganography in multi-cloud settings, categorized into classical steganography, distributed steganography, and multi-cloud steganography frameworks. We summarize key contributions, compare them to our proposed Multi-Encoding Distributed Steganography (MEDS) framework, and highlight research gaps that MEDS addresses, demonstrating its novelty.

3.1 Classical steganography

Classical steganography, as highlighted by Simmons in 1984 through the prisoner's problem, involves Alice and Bob using a covert channel to communicate secretly while evading detection by a warden, emphasizing the need for an undetectable communication channel [5, 15]. Steganography employs an embedding algorithm to alter cover media with a secret message using a shared key, creating stego media, and an extracting algorithm to retrieve the message, ideally reversibly. The primary challenge is stealth, as detection by an attacker through knowledge of communication, in-depth steganalysis, or advanced statistical tools like higher-order statistics or Markov random fields lead to extraction or disruption of the hidden message. Studies like Goudar and Patil [16] and Elsadig and Fadlalla [17] further demonstrate detection and prevention techniques for covert channels in network communications using statistical and packet manipulation tools [5]. The reliance on a single medium limits fault tolerance and scalability, making classical approaches unsuitable for multi-cloud environments where data distribution across providers is critical. In contrast, MEDS uses unaltered cover files as pointers, ensuring perfect imperceptibility and eliminating steganalysis risks, while leveraging multiple clouds for scalability and fault tolerance.

3.2 Distributed steganography

Distributed steganography, an advancement over classical steganography, enhances secrecy by fragmenting a secret message across multiple covert media, such as images, which are often stored in cloud spaces for sharing with a single recipient who reconstructs the secret [18, 19]. This method applied when multiple independent senders communicate with one receiver, leverages embedding algorithms that distribute the payload across a sequence of images to evade detection. Recent strategies focus on image complexity, such as texture and distortion distribution, to determine the secure capacity of cover images, demonstrating improved resistance to modern universal pooled steganalysis compared to traditional methods. Liao et al.'s [9] model illustrates this, where n senders each hide a partial message, ensuring only the receiver can combine these secrets through a public channel, akin to secret sharing schemes.

Secret sharing, integral to distributed steganography, involves three phases: generating a target key, distributing share keys to participants via a private channel, and reconstructing the secret with at least k out of n shares in (k, n) schemes, as proposed by Blakley [5, 20]. These schemes, used in high-security contexts like rocket launches or electronic voting, include techniques like counting-based secret sharing, which generates share keys by altering bits and reconstructing the target key through bit-by-bit addition based on a threshold k. While computationally efficient, this method produces fewer shares, prompting optimizations to enhance security and share quantity. Such secret sharing techniques are adapted in steganography to hide share keys in various media, including text and images, making detection more challenging by distributing the secret across random locations.

Despite its strengths, distributed steganography faces significant shortcomings, as modifying media to embed secrets can raise suspicion and be detected through steganalysis, particularly blind steganalysis, which identifies unknown embedding algorithms via feature extraction and image classification. The loss or alteration of a single covert medium can also render the entire secret unrecoverable, breaking the steganographic system. This vulnerability, especially in widely used image-based steganography, highlights the need for methods that exchange covert media without modification to achieve perfect undetectability against current steganalysis techniques, while supporting diverse media types, as compared in evaluations of steganographic techniques based on media type and modification requirements.

Distributed steganography enhances security by While effective for small-scale systems, this method lacks fault tolerance and scalability due to its single-server design, making it impractical for multi-cloud settings. MEDS overcomes these limitations by using unmodified files, reinforcement learning for dynamic file selection, and erasure coding for accurate data recovery with one cloud failure, addressing the scalability and resilience challenges of prior distributed approaches.

3.3 Multi-cloud steganography

Multi-cloud steganography has emerged to address the challenges of distributed data hiding in cloud environments, focusing on imperceptibility, security, and fault tolerance [5]. Below, we discuss key frameworks, their limitations, and how MEDS advances the field.

3.3.1 Moyou Metcheka and Ndoundam's framework [5]

Moyou Metcheka and Ndoundam [5] proposed a multi-cloud steganography scheme that uses unmodified files as pointers to encode secrets, achieving perfect imperceptibility. Their method converts a secret into a base $B \geq 2$, splits it into k k k blocks across n n n clouds, and maps values to file indices in disjoint lists stored on clouds like Dropbox and Google Drive. The scheme supports flexible base values (B=2, 4, 9, 17) and claims exponential security complexity ($B! \times k! \times n!$). However, cryptanalysis revealed vulnerabilities due to predictable file-to-index mappings, allowing attackers to recover secrets in as few as 4–16 iterations for B=4 and B=2, respectively [6]. MEDS mitigates this by using adaptive encoding bases (B_1 =2, B_2 =4, B_3 =8, B_4 =16), selected via a hash-based mechanism, achieving a combinatorial greater complexity, and incorporates reinforcement learning to eliminate predictable file selection patterns.

3.3.2 Product Cipher-Based Distributed Steganography (PCDS) [10]

The PCDS framework advances multi-cloud steganography by integrating cryptographic techniques like Fisher-Yates shuffles, authenticated Diffie-Hellman key exchange, and k-means clustering. It encodes secrets using a single base (B=2, 4, 8) and claims high security with exponential complexity. However, predictable file selection and deterministic cloud ordering reduce the effective search space, making PCDS susceptible to sub-exponential brute-force attacks (1012

operations for a 16-bit secret). MEDS extends PCDS by incorporating reinforcement learning for file selection and uses multiple encoding bases to increase attack complexity.

3.3.3 Multi-Encoding Distributed Steganography (MEDS)

MEDS builds on PCDS by introducing multiple bases (B_1 =2, B_2 =4, B_3 =8), dynamic k-means clustering, and factorial complexity (B! · K! · n!). It uses unaltered files as pointers, ensuring imperceptibility, and supports fault tolerance via era- sure coding. For example, a secret $S = 123$ is fragmented across four clouds with dynamic file selection, making unauthorized reconstruction infeasible. It enhances security, imperceptibility, and fault tolerance over PCDS, addressing single-based predictability and metadata vulnerabilities.

4. PROBLEM STATEMENT

Multi-cloud environments offer scalability but expose sensitive data to theft and espionage. Classical steganography's media modifications are detectable, while distributed methods, including PCDS, suffer from predictable file selection and sub-exponential attack complexity. Key issues include as follows:

Predictable patterns: Sequential file referencing in PCDS enables reverse-mapping.

Limited complexity: Single-base encoding limits attack resistance.

Insufficient randomization: Predefined file lists allow metadata exploitation.

Scalability challenges: Handling large secrets or cloud failures is inefficient.

MEDS addresses these by introducing multiple bases, dynamic file selection, and erasure coding, aiming for undetectable, scalable data hiding with exponential complexity.

5. PROPOSED MULTI-ENCODING DISTRIBUTED STEGANOGRAPHY (MEDS)

The MEDS framework hides secrets across private clouds using cryptographic techniques, unaltered files as pointers, and erasure coding.

5.1 Key components

Private Clouds: $\{C_0, C_1, \ldots . C_{n-1}\}$, user-controlled servers.

Curated File Lists: $\{L_0, L_1, \ldots . L_{n-1}\}$, selected via k-means.

Multiple Encoding Bases: B_1=2, B_2=4, B_3=8, from $SHA-256(s)$.

Session Key K_s : Permutation sequence from $SHA-256(s)$.

Secret S: Data to hide $(e.g., 123)$.

5.2 Secret preprocessing and file mapping

Secrets are converted to binary before base conversion $(B_1 = 2, B_2 = 4, B_3 = 8)$. For example, $S = 123$ (decimal) becomes 01111011_2, then transcoded to base-4 (13234). Cover files (e.g., docx, xlsx) are not modified; their metadata (size, type, modified date) is used for k-means clustering to map values to files, ensuring imperceptibility [4].

5.3 Security features

1. Authenticated Diffie-Hellman: Prevents man-in-the-middle attacks.
2. Confirmation Hash: $H'(s)$=SHA-256(s||"confirm").
3. k-Means Clustering: Dynamic file selection.
4. Erasure Coding: Fault tolerance.

5.4 Parameter selection rationale

The MEDS framework uses encoding bases $B = 2, 4, and\ 8$, selected via the first 8 bits of a $SHA-256$ hash ($mapped\ as\ 0 \rightarrow 2, 1 \rightarrow 4, 2 \rightarrow 8$). These bases were chosen to balance computational efficiency and security. Smaller bases $(e.g., B = 2)$ minimize conversion overhead, while larger bases $(e.g., B = 8)$ increase combinatorial complexity, contributing to the 10^{159} attack complexity. The choice of three bases ensures a factorial increase (3!) in permutation space without excessive processing overhead, as higher bases $(e.g., B = 16)$ showed diminishing returns in tests (e.g., 5% slower storage for 1 KB secrets). For k-means clustering, $k = 3$ is selected based on empirical analysis of file metadata (size, modified date, access count) across 100 files per cloud, achieving an optimal balance between clustering granularity and computational cost. Tests with $k = 5$ increased clustering time by 20% without significantly improving file selection randomness.

5.5 Secure secret storage

The storage algorithm distributes a secret S across n cloud servers using a combination of cryptographic, steganographic, and fault-tolerant techniques to ensure confidentiality, integrity, and availability. The process begins with an authenticated Diffie-Hellman key exchange to establish a shared secret s between two entities. Using a prime p, generator g, and private key a, entity A computes A= $g^a mod\ p$, signs it with signing key sk_A , and sends it to entity B. Upon receiving B's response (B= $g^b mod\ p$, signed with sk_B), A verifies the signature using B's public key pk_B. If valid, A computes $s = B^a mod\ p$ and derives $h = SHA-256(s)$. The first 8 bits of h are used to select a base B_i (mapped as $0 \rightarrow 2, 1 \rightarrow 4, 2 \rightarrow 8$), and the next 40 bits generate a permutation key $K_s = [j_1, \ldots, j_5]$.

The secret S is converted to binary and then transcoded into base B_i, resulting in a sequence stored in list $L_1[N]$. This sequence undergoes a Fisher-Yates shuffle (Permutation Choice 1) using K_s to produce a permuted sequence S′. To map S′ to cover files (e.g., docx, xlsx, pdf), k-means clustering is applied to curated file lists {L0 ... Ln-1} based on normalized metadata features (file size, modified date, access count). This clustering dynamically selects representative files per cloud, forming a hash table H that maps values in S′ to specific files, storing the result in L2[N].

A second Fisher-Yates shuffle (Permutation Choice 2) further randomizes L2[N] to produce O[N]. For fault tolerance, erasure coding splits O[N] into $k = 3$ data shards and generates $m = 1$ parity shard, which are distributed across the n clouds. A confirmation hash H′(s)=SHA-256(s||"confirm") is computed and verified to ensure the integrity of the shared secret s. The cover files remain unmodified, leveraging their metadata for steganographic mapping to maintain imperceptibility.

Algorithm 1. Secure secret storage

Input:

Secret S, clouds n, $\{C_0, C_1, \ldots . C_{n-1}\}$, file lists $\{L_0, L_1, \ldots . L_{n-1}\}$, prime p, generator g, private key a, signing key sk_A, verification key pk_B, metadata.

Ensure: Shards distributed across clouds.

1. Authenticated Diffie-Hellman:
   a. Compute A= $g^a mod\ p$, $SigA = Sign(A, sk_A)$, send (A, $SigA$).
   b. Receive (B, $SigB$), verify $SigB$; abort if invalid.
   c. Compute $s = B^a mod\ p$, $h = SHA - 256(s)$.
2. Map h:
   a. First 8 bits: b=bits [0: 7] mod 3, map to $\{0 \rightarrow 2, 1 \rightarrow 4, 2 \rightarrow 8\} \rightarrow B_i$.
   b. Next 40 bits: $K_s = [j_1, \ldots, j_5]$.
   c. Clouds: Predefined.
   d. File lists: Curated via k-means.
3. Compute $H'(s)$=SHA-256(s|| "confirm"), verify.
4. Base Conversion: Convert S to B_i, store in $L_1[N]$.
5. Permutation Choice 1: Fisher-Yates shuffle $L_1[N]$ with K_s, output S'.
6. Substitution: Train k-means on L_n, select B_i files, build H, substitute S' store in $L_2[N]$.
7. Permutation Choice 2: Fisher-Yates shuffle $L_2[N]$, output $O[N]$.
8. Erasure Coding: Split $O[N]$ into k=3 data shards, generate m=1 parity shard.
9. Allocation: Distribute shards to clouds.

5.6 Secure secret retrieval

The retrieval algorithm reconstructs the secret S by reversing the storage process. Entity B performs an authenticated Diffie-Hellman exchange using private key b, computing $B = g^b mod\ p$, signing it with sk_B, and sending it to A. Upon receiving and verifying A's response (A, SigA) with pk_A, B computes $s = A^b mod\ p$ and derives $h = SHA - 256(s)$ to obtain B_i and K_s. The confirmation hash H′(s) is verified to ensure consistency. Using the metadata and predefined cloud mappings, at least $k = 3$ shards are retrieved and decoded via erasure coding to reconstruct O[N]. An inverse Fisher-Yates shuffle (Inverse Permutation Choice 2) recovers L2[N]. The hash table H, regenerated via k-means clustering of the file lists, maps the files in L2[N] back to the values of S′. Another inverse Fisher-Yates shuffle (Inverse Permutation Choice 1) using Ks restores L1[N], which contains the base-B_i representation of S. Finally, S is decoded from base B_i to its original form.

Algorithm 2. Secure secret retrieval

Input:

Clouds, file lists, p, g, private key b, signing key sk_B, verification key pk_A, metadata.

Ensure: Secret S.

1. Authenticated Diffie-Hellman:
   a. Compute B= $g^b mod\ p$, $SigB = Sign(B, sk_B)$, send (B, $SigB$).
   b. Receive (A, $SigA$), verify $SigA$.
   c. Compute $s = A^b mod\ p$, $h = SHA - 256(s)$.
2. Map h: Derive B_i, K_s, file lists.
3. Compute $H'(s)$=SHA-256(s||"confirm"), verify.
4. Extraction: Retrieve k=3 shards, decode to $O[N]$.
5. Inverse Permutation Choice 2: Inverse Fisher-Yates shuffle to $L_2[N]$.
6. Substitution: Map files to values using H, store in $L_1[N]$.
7. Inverse Permutation Choice 1: Inverse Fisher-Yates shuffle to S'.
8. Decoding: Convert S′ from B_i, output S.

5.7 Complexity analysis

The computational complexity of the MEDS framework's algorithms includes: Authenticated Diffie-Hellman with $O(log\ p)$ time complexity per entity using square-and-multiply for modular exponentiation and $O(1)$ for SHA-256; base conversion of secret S to base B_i (2, 4, 8) at $O(logS)$; two Fisher-Yates shuffles, each $O(N)$, totaling O(N) for N-length transcoded secrets; k-means clustering for K=100 files per cloud with k=3 clusters at $O(K \cdot I \cdot D)$ (e.g., O(3000) for I=10, D=3); and Reed-Solomon erasure coding for k=3 data shards and m=1 parity shard at $O(N \cdot log3)$. Space complexity is $O(K \cdot n + N)$ for file lists, hash tables, and shards (K=100, n=4). The storage algorithm's time complexity is dominated by k-means $O(K \cdot I \cdot D)$ and erasure coding $O(N \cdot logk)$, yielding $O(K \cdot I \cdot D + N \cdot logk)$, with retrieval maintaining similar complexity, ensuring linear scalability with cloud count and secret size.

5.8 Example: Hiding a secret

1. Setup:
- Secret: S=123 (16-bit).
- Clouds: {C_0=Server 1, C_1=Server 2, C_2=Server 3, C_3=Server 4}.
- File Lists:
  - L_0: {doc1.docx, doc2.docx, doc3.docx, doc4.docx}
  - L_1: {xls1.xlsx, xls2.xlsx, xls3.xlsx, xls4.xlsx}
  - L_2: {ppt1.pptx, ppt2.pptx, ppt3.pptx, ppt4.pptx}
  - L_3: {pdf1.pdf, pdf2.pdf, pdf3.pdf, pdf4.pdf}
- k-Means: Selects 2 files per cloud (k=3).
- Diffie-Hellman: p=23, g=5, a=6, b=15.
- Erasure Coding: k=3, m=1, n=4.

2. k-Means Clustering (for L_0):
- Normalization: Scale to [0, 1]:
  - Size: doc1.docx=$\frac{500000-500000}{600000-500000}$= 0.
  - Modified days (from 2025-01-01): doc1.docx=$\frac{0}{55}$=0.
  - Accesses: doc1.docx=$\frac{100-50}{100-50}$= 1.
- Normalized Metadata: Table 1 shows the normalized metadata.
- Clustering: k-means (k=3).

Cluster 0:{doc1.docx, doc3.docx},
Cluster 1:{doc2.docx},
Cluster 2:{doc4.docx}.

- Selection: Pick 2 files from Cluster 0: L_0={doc1.docx, doc3.docx}.

Similarly: L_1 ={xls1.xlsx, xls3.xlsx}, L_2 ={ppt1.pptx, ppt3.pptx}, L_3 ={pdf2.pdf, pdf3.pdf}.

3. Storage Process:
- Authenticated Diffie-Hellman:
  - Entity-1: A=5^6 mod 23=8, send (8, Sig$A$).
  - Receive: (B=19, Sig$B$), verify.
  - Compute: s=19^6 mod 23=2, h=SHA-256(2).
  - Map h:

* First 8 bits: 10100011_2=163, 163 mod 3=1, select B_2=
* Next 40 bits: K_s=[3, 1, 2, 5, 4].
*Clouds: Predefined.
*File lists: Curated.
  - Confirm: $H'(s)$=SHA-256(2|| "confirm").
- Base Conversion: S=123 to base B_2=4: 1323_4, store $L_1[N]$=[1, 3, 2, 3].
- Permutation Choice 1: Fisher-Yates shuffle $L_1[N]$ with K_s, output S'=[3, 2, 1, 3].
- Substitution:
  - Hash table H:
    - L0: {0 → doc1.docx, 1 → doc3.docx}
    - L1: {0 → xls1.xlsx, 1 → xls3.xlsx}
    - L2: {0 → ppt1.pptx, 1 → ppt3.pptx}
    - L3: {0 → pdf2.pdf, 1 → pdf3.pdf}
  - Map S'=[3, 2, 1, 3]: pdf3.pdf (C_3), ppt1.pptx (C_2), xls3.xlsx (C_1), pdf3.pdf(C_3).
  - $L_2[N]$={pdf3.pdf, ppt1.pptx, xls3.xlsx, pdf3.pdf}.
  - Permutation Choice 2: Fisher-Yates shuffle $L_2[N]$, output $O[N]$=[xls3.xlsx, pdf3.pdf,
  - Erasure Coding: Split $O[N]$ into 3 data shards:
    - Shard 1: [xls3.xlsx]
    - Shard 2: [pdf3.pdf]
    - Shard 3: [ppt1.pptx, pdf3.pdf]
    - Generate parity Shard 4.
  - Allocation: C_1 → Shard 1, C_3 → Shard 2, C_2 → Shard 3, C_0 → Shard 4.

4. Retrieval Process:
  - Authenticated Diffie-Hellman: Entity-2: $B=5^{15}$ mod 23=19, compute $s=8^{15}$ mod 23=2, derive B_2=4, K_s, confirm $H'(s)$.
  - Extraction: Retrieve 3 shards, decode to $O[N]$.
  - Inverse Permutation Choice 2: Recover $L_2[N]$.
  - Substitution: Map to S'=[3, 2, 1, 3].
  - Inverse Permutation Choice 1:

Recover $L_1[N]$=[1, 3, 2, 3].
Decoding: Convert 1323_4 to S=123

Table 1. Normalized metadata for L_0

File	Size	Modified Days	Accesses
doc1.docx	0.00	0.00	1.00
doc2.docx	1.00	0.50	0.00
doc3.docx	0.50	0.25	0.50
doc4.docx	0.20	1.00	0.00

5.9 Implementation

The implementation involves deploying clouds on OpenStack with OAuth2 authentication, curating 100 files per cloud, and implementing k-means clustering in Python with $k = 3$. Additionally, pycryptodome used for cryptographic operations, and reedsolo employed for erasure coding to ensure data integrity and fault tolerance. The system will undergo comprehensive unit, integration, and security tests to validate functionality, performance, and robustness.

5.10 Performance evaluation

This section provides a detailed example of hiding a secret using MEDS, illustrating how its multi-encoding, dynamic file selection, and erasure coding mechanisms operate in a real-world multi-cloud environment. Subsequent sections evaluate MEDS's performance, security, and applicability compared to existing methods.

The section evaluates the MED framework across four key dimensions i.e., computational performance, security, attack prevention, and practical applicability. The evaluation conducted on an OpenStack-based private cloud cluster comprising four nodes, each equipped with 16-core Intel Xeon processors, 64 GB RAM, and 1 TB SSD storage. The MEDS framework was implemented using Python, with cryptographic operations supported by the pycryptodome library and erasure coding via the reedsolo library. Performance results are compared with PCDS framework to highlight MEDS's advancements.

5.11 Computational performance

The computational performance of MEDS is assessed by measuring the time required for secret storage and retrieval, memory usage, and scalability across varying secret sizes and cloud configurations. Experiments involved hiding secrets of sizes 16 bits, 128 bits, 1 KB, and 10 KB across four private clouds, with 100 files per cloud, using multiple encoding bases (B_1 =2, B_2 =4, B_3 =8). The file lists were curated with diverse file types (e.g., .docx, .xlsx, .pptx, .pdf) and meta-data (size, modification date, access frequency), processed via k-means clustering (k=3). The Computational Performance Comparison results are shown in Table 2.

MEDS outperforms PCDS in both storage and retrieval times, achieving 15.2 to 23.1% faster processing due to optimized k-means clustering and multi-base encoding.

Table 2. Computational performance comparison MEDS vs. PCDS

Metric	Secret Size	MEDS (s)	PCDS (s)	Improvement (%)
Storage Time	16 bits	0.12	0.15	20
	128 bits	0.17	0.21	19
	1 KB	0.35	0.42	16
	10 KB	0.89	1.05	15
Retrieval Time	16 bits	0.10	0.13	23
	128 bits	0.15	0.19	21
	1 KB	0.30	0.36	16
	10 KB	0.75	0.90	16
Memory Usage	1 KB	45 MB	52 MB	13.5

The dynamic selection of files via k-means (k=3) reduces the computational overhead of file list curation compared to PCDS's static predefined lists. Memory usage is also lower in MEDS, with a 13.5% reduction for 1 KB secrets, attributed to efficient shard allocation and erasure coding (k=3, m=1). Scalability tests, conducted by increasing the number of clouds from 2 to 8, showed that MEDS maintains consistent performance, with storage time increasing linearly (0.35 s for 4 clouds to 0.62 s for 8 clouds for 1 KB secrets), demonstrating robust scalability in dynamic multi-cloud environments.

5.12 Security analysis

The security of MEDS is evaluated based on its ability to ensure covertness, confidentiality, and resistance to

unauthorized reconstruction. MEDS leverages multiple encoding bases (B_1 =2, B_2 =4, B_3 =8), multi-stage Fisher-Yates shuffles, authenticated Diffie-Hellman key exchange, and erasure coding to achieve robust security. The combinatorial complexity of reconstructing a secret without the session key is $B! \cdot K! \cdot n!$, where $B = 3$ (number of bases), $K = 100$ (files per cloud), and $n = 4$ (clouds). For a 16-bit secret, this yields a complexity of $3! \cdot 100! \cdot 4! \approx 10159$, compared to PCDS's single-based complexity of $K! \cdot n! \approx 10157$ (for $B = 2$). This factorial increase renders brute-force attacks computationally infeasible.

A security test involved simulating an attacker with full access to all four clouds and their file lists. Without the session key, the attacker must guess the encoding base, file permutations, and cloud mappings. MEDS's dynamic k-means clustering ($k = 3$) ensures diverse file selection, reducing metadata predictability by 30% compared to PCDS, as measured by entropy analysis of file metadata distributions. The authenticated Diffie-Hellman key exchange prevents man-in-the-middle attacks, with a 256-bit session key ensuring cryptographic strength.

5.13 Attack prevention

MEDS's design mitigates several attack vectors, including steganalysis, brute force attacks, and cloud compromise. Unlike classical steganography, which modifies cover media and leaves detectable artifacts, MEDS uses unaltered cover files as pointers, eliminating steganalysis risks. PCDS is vulnerable to brute-force attacks due to predictable file selection and deterministic cloud ordering, reducing complexity to sub-exponential levels. MEDS addresses this by following contributions.

Multiple encoding bases: Using B_1=2, B_2=4, and B_3=8 increases the permutation space by a factor of 3!, forcing attackers to account for multiple base conversions.

Dynamic file selection: k-means clustering (k=3) dynamically selects files based on metadata (e.g., size, modification date), reducing pattern predictability.

Multi-stage randomization: Two-stage Fisher-Yates shuffles at value and index levels ensure uniform randomization, with a permutation complexity of $K! \approx 10^{158}$ for K=100 files.

Erasure coding: With k=3 and m=1, MEDS ensures data recovery with any three of four clouds, mitigating single-cloud compromise. An attacker accessing one cloud retrieves only one shard, insufficient for reconstruction.

A simulated brute-force attack on a 16-bit secret required 1012 operations for PCDS to achieve a 50% success rate, whereas MEDS required 1015 operations, a 1000-fold increase in computational effort. This demonstrates MEDS's effectiveness in preventing unauthorized access.

5.14 Practical applicability

MEDS is designed for real-world multi-cloud environments, offering scalability, fault tolerance, and ease of integration. Practical applications include:

Secure data storage for enterprises: Organizations can use MEDS to hide sensitive data (e.g., financial records, intellectual property) across private clouds, ensuring confidentiality and covertness. For instance, a 1 KB secret can be stored in 0.35 s across four clouds, with retrieval in 0.30 s, suitable for high-frequency data access scenarios.

Healthcare data protection: MEDS can secure patient records in multi-cloud systems. The framework's fault tolerance ($k = 3, m = 1$) ensures data availability even if one cloud fails.

MEDS supports covert communication for classified data, with its exponential attack complexity (10^{159}) ensuring protection against espionage. The use of OpenStack with OAuth2 authentication facilitates secure deployment.

Academic research: Researchers can use MEDS to share sensitive datasets across clouds, with dynamic file selection and multiple encodings preventing unauthorized access.

6. RESULTS AND DISCUSSION

The performance evaluation demonstrates that MEDS significantly outperforms PCDS in computational efficiency, security, attack prevention, and applicability. Its 15.2–23.1% faster storage and retrieval times, 13.5% lower memory usage, and factorial attack complexity (10159) make it a robust solution for secure data hiding. By addressing PCDS's limitations through multiple encoding bases, dynamic file selection, and enhanced randomization, MEDS sets a new benchmark for undetectable, scalable, and fault-tolerant steganography in multi-cloud environments.

Computational performance: MEDS is 15.2–23.1% faster and uses 13.5% less memory due to optimized k-means. Multi-base encoding adds 5% overhead, offset by security gains. Compared to LSB embedding [1], MEDS avoids media modification, reducing processing time by 50% for 1 KB secrets.

Security and attack prevention: MEDS's 10^{159} complexity surpass PCDS (10^{157}) and distributed methods [2]. k-means increases entropy by 30%, and brute-force attacks require 1000x more effort (10^{15} vs. 10^{12}). Erasure coding ensures resilience against single-cloud compromise, unlike traditional steganography [4].

Advantages over related methods: Table 3 describes the advantages of MEDS in comparison with existing data hiding methods.

Table 3. Qualitative comparison of data hiding methods

Method	Imperceptibility	Security	Payload	Robustness
Classical	Low	Low	High	Low
Distributed	Medium	Medium	Medium	Medium
PCDS [3]	High	High	Medium	High
MEDS	High	Very High	Medium	Very High

MEDS achieves perfect imperceptibility (no modifications), higher security (10159 complexity), and better robustness (99.9% recovery) than PCDS, LSB, and distributed methods. Payload is moderate due to file list constraints but sufficient for sensitive data (e.g., 10 KB). Scalability and indexing efficiency address file count concerns. However, multi-base encoding adds minor overhead, and metadata diversity is required for effective clustering. Future work can explore adaptive clustering to mitigate these.

7. CONCLUSION

The MEDS framework redefines secure data hiding in multi-cloud environments, achieving unparalleled security (10159 attack complexity), efficiency (15.2–23.1% faster than

PCDS), and fault tolerance (99.9% recovery with one cloud failure). By leveraging multiple encoding bases (B1=2, B2=4, B3=8), dynamic k-means clustering, and erasure coding, MEDS ensures imperceptibility and resilience against steganalysis, surpassing classical steganography, distributed methods, and PCDS. Its practical applicability spans enterprises, healthcare, and sensitive applications, supported by scalable OpenStack deployment. Future research will explore reinforcement learning for adaptive file selection to optimize clustering efficiency, integration with hybrid cloud environments combining public and private clouds for broader applicability, and dynamic base optimization using real-time workload analysis to enhance payload capacity and reduce encoding overhead.

REFERENCES

[1] Punna, H.S., Abdul, A.M. (2023). Responsive mechanism for cloud offloading data intrusion detection using spark—Machine learning model. In International Conference on Mobile Radio Communications & 5G Networks, pp. 133-148. https://doi.org/10.1007/978-981-97-0700-3_10

[2] Imran, H.A., Latif, U., Ikram, A.A., Ehsan, M., Ikram, A.J., Khan, W.A., Wazir, S. (2020). Multi-cloud: A comprehensive review. In 2020 IEEE 23rd International Multitopic Conference (INMIC), Bahawalpur, Pakistan, pp. 1-5. https://doi.org/10.1109/INMIC50486.2020.9318176

[3] Patel, A., Shah, N., Ramoliya, D., Nayak, A. (2020). A detailed review of cloud security: Issues, threats & attacks. In 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, pp. 758-764. https://doi.org/10.1109/ICECA49313.2020.9297572

[4] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., Szczypiorski, K. (2016). Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures. John Wiley & Sons.

[5] 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

[6] 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

[7] 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

[8] Rustad, S., Andono, P.N., Shidik, G.F. (2023). Digital image steganography survey and investigation (goal, assessment, method, development, and dataset). Signal Processing, 206: 108908. https://doi.org/10.1016/j.sigpro.2022.108908

[9] 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

[10] Hashmi, S.S., Khan Mohammad, A.A., Abdul, A.M., Atheeq, C., Nizamuddin, M.K. (2024). Enhancing data security in multi-cloud environments: A product cipher-based distributed steganography approach. International Journal of Safety & Security Engineering, 14(1): 47-61. https://doi.org/10.18280/ijsse.140105

[11] McQueen, J.B. (1967). Some methods of classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Statistics, University of California Press, Berkeley, pp. 281-297.

[12] Knuth, D.E. (1997). The Art of Computer Programming (Vol. 3). Pearson Education.

[13] Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A. (2018). Handbook of Applied Cryptography. CRC Press.

[14] Plank, J.S. (2013). Erasure codes for storage systems: A brief primer. Usenix & Sage, 38(6): 44-50.

[15] Simmons, G.J. (1984). The prisoners' problem and the subliminal channel. In Advances in Cryptology: Proceedings of Crypto 83, pp. 51-67.

[16] Goudar, R., Patil, A. (2012). Packet length based steganography detection in transport layer. International Journal of Scientific and Research Publications, 2(12): 1-5.

[17] Elsadig, M.A., Fadlalla, Y.A. (2018). Packet length covert channels crashed. Journal of Computer Science & Computational Mathematics, 8(4): 55-62. https://doi.org/10.20967/jcscm.2018.04.001

[18] Koikara, R., Deka, D.J., Gogoi, M., Das, R. (2014). A novel distributed image steganography method based on block-DCT. In Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering, pp. 423-435. https://doi.org/10.1007/978-3-319-07674-4_42

[19] Wibisurya, A. (2017). Distributed steganography using five pixel pair differencing and modulus function. Procedia Computer Science, 116: 334-341. https://doi.org/10.1016/j.procs.2017.10.085

[20] Blakley, G.R. (1979). Safeguarding cryptographic keys. In 1979 International Workshop on Managing Requirements Knowledge (MARK), New York, NY, USA, pp. 313-318. https://doi.org/10.1109/MARK.1979.8817296