

An Improved Mechanism to Maintain Data Integrity and Anomaly Detection in Cloud Storage

¹Aditi Joshi, ²Rajendra Prasad Mahapatra and ³Ganesh Gopal Devarajan

^{1,2}Department of Computer Science and Engineering, Faculty of Engineering and Technology,

SRM Institute of Science and Technology, Delhi, NCR Campus, Modinagar, Ghaziabad, Uttar Pradesh, India.

³Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh, India.

¹aj7501@srmist.edu.in, ²mahapatra.rp@gmail.com, ³dganeshgopal@gmail.com

Correspondence should be addressed to Ganesh Gopal Devarajan : dganeshgopal@gmail.com

Article Info

Journal of Machine and Computing (<https://anapub.co.ke/journals/jmc/jmc.html>)

Doi : <https://doi.org/10.53759/7669/jmc202606022>

Received 17 June 2025; Revised from 13 August 2025; Accepted 22 September 2025.

Available online 25 October 2025.

©2026 The Authors. Published by AnaPub Publications.

This is an open access article under the CC BY-NC-ND license. (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Abstract— There are different services which can be offered by cloud platforms such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). These services and resources are provided by the third party and available on-demand through the Internet. It enables the organizations to focus on the infrastructure and services rather than maintain it. It includes several challenges such as vendor lock-in, security, data integrity, and compliance issues. To address these issues is very challenging for organizations and relies on cloud services. However, data integrity is one of the critical challenges due to data inconsistency, user trust, completeness, and compliance. Therefore, to ensure the data integrity distinct mechanisms can be applied such as encryption, access control, regular audits, backups, and data validations. The integration of these strategies results in a comprehensive framework for maintaining data integrity in cloud storage, ensuring trustworthiness, security, and compliance with regulatory standards. The result set indicates the higher accuracy approximate 1.0 with good number of instances of benign and suspicious data. The major aim of the current research is to enhance the security of cloud data by using AES algorithms and provide double encryption. The extraction of the top features by using recursive feature elimination. Finally, AI-powered anomaly detection is proposed for data integrity and to detect anomalous access patterns, unauthorized changes, and suspicious transactions in real-time, helping prevent integrity violations before they escalate.

Keywords – Cloud Computing, BETH Dataset, Anomaly Detection, Machine Learning, Stream Learning.

I. INTRODUCTION

Next- The generation architecture of IT organizations is cloud computing, which offers a number of features never seen in IT history, such as on-demand, ubiquitous network connectivity, location-independent resource pooling, quick resource flexibility, usage-based pricing, and risk transference. Cloud Computing, disturbing technology with the main consequences, transforms how businesses use information technology. Outsourcing or centralizing data to the cloud is one of the primary characteristics of this paradigm shift. Individuals and businesses say that data storage remotely in the cloud flexible on request offers tempting benefits, such as reducing storage burden, allowing universal access to data from various geographic locations and avoiding capital expenditures for employees, software and hardware maintenance. Although these benefits of cloud computing are undeniable, cloud customers may not be aware of the internal workings of cloud service providers (CSP) because of the cloud's opaqueness as distinct administrative entities. Users who outsource their data also give up ultimate control over what happens to their data [1]. Because of its elasticity and low maintenance costs, cloud storage is becoming more and more popular. It provides an on-demand data outsourcing service model. However, outsourcing data storage to outside cloud storage providers raises security issues. If cloud clients' data is inadvertently corrupted or intentionally infiltrated by insider or outer assaults, it is preferable to allow them to confirm the integrity of the data they have outsourced. Cloud storage is also important for long-term archiving, which is a task that is written once and rarely retrieved. Whole-file inspection becomes prohibitive due to the frequent enormous volume of preserved data,

even though the saved data is rarely retrieved. Maintaining its integrity is therefore essential for disaster recovery or legal compliance [2]. Cloud computing architecture allows the user to access the necessary data more quickly by allowing many cloud components to communicate with one another about the different types of data they are storing (**Fig 1**). Cloud computing is mainly concerned with the front end and back end. While the many data storage devices and servers that make up the cloud constitute the backend, the user that needs the data is the front end.

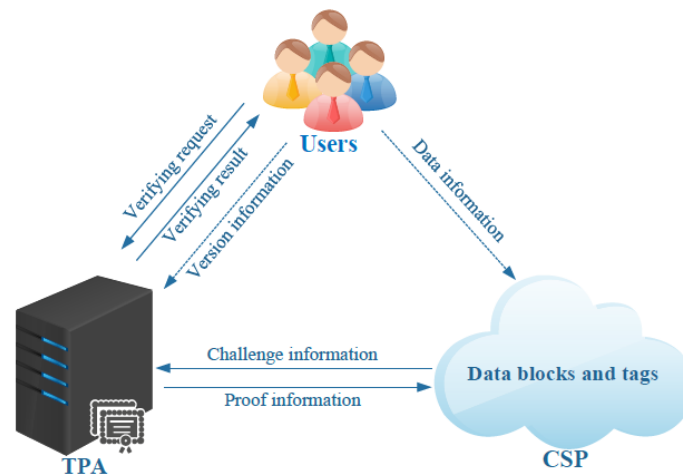


Fig 1. Secure Cloud Storage Model [3].

Cloud storage benefits service providers as well as users. Cloud users save money and are freed from the burden of maintaining their infrastructure [1]. Cloud storage gives a user access to services regardless of where they are [4]. By distributing the existing infrastructure across several users, a CSP might generate substantial money. The optimal use of energy is the result of this high degree of resource utilization. Even with these advantages, cloud storage has some inherent drawbacks because CSPs are not always trusted and may have nefarious intentions. The malevolent behavior of a CSP can be either purposeful or inadvertent (the service provider is not aware whether corrupt sectors are produced in the disk or the hard disk fails). Some of the specific user concerns with cloud storage include stability, data lock-in, unauthorized physical access, and cloud storage provider espionage [4]. The cloud storage allows data to spread on multiple servers and geographical locations. The user no longer has control of his data and is unable to visually explore the data interconnection. One risk associated with the cloud warehouse is therefore an unauthorized physical approach. Re-evaluation of storage space, elimination of obsolete equipment or reuse of units may result in data disruption. Consumers often fear that service providers will try to obtain personal data from them without their consent. There are no established guidelines for how data should be kept up to date and saved on the cloud. For storing data, each CSP has a proprietary format. As a result, customers are locked into a specific CSP and find it difficult to move to another. Regular modifications to cloud-based services also lead to issues for consumers.

Despite its numerous advantages such as scalability, cost-efficiency, and accessibility, faces several challenges that impact its reliability, security, and performance. Data security and privacy pose significant challenges due to storing sensitive information on third-party servers, which increases the risk of security breaches, unauthorized access, and cyberattacks. While encryption methods can mitigate these risks to some extent, managing access control and encryption keys remains a complex task. Ensuring data integrity, which involves maintaining the consistency and accuracy of data over time, presents another major challenge [5]. The authenticity of data stored in the cloud may be compromised by intentional tampering, unauthorized alterations, or inadvertent corruption. Techniques like hash-based verification, merkle trees, and blockchain-based validation can improve integrity, but they demand more computational resources. Latency and performance issues can be quite challenging, particularly when dealing with massive datasets or real-time applications. Ensuring data integrity in cloud storage is a crucial challenge that impacts the dependability, precision, and uniformity of data stored in remote locations. Cloud storage systems are susceptible to a range of risks, such as unauthorized changes, accidental corruption, and deliberate tampering, which can undermine the reliability of stored data. One of the main worries is unauthorized access and data breaches, where cybercriminals take advantage of weaknesses in cloud security to manipulate or delete important data. In the absence of robust authentication and encryption measures, malicious actors can manipulate data, resulting in inconsistencies and a loss of trust in cloud services. One significant challenge is data corruption caused by hardware malfunctions, software bugs, or transmission errors [6]. Cloud storage depends on multiple data centres and interconnected networks, where data travels through various nodes, increasing the likelihood of corruption during transmission or storage. Malicious insider threats are a concern, as employees or third-party service providers with access to cloud systems may intentionally alter or delete data for personal gain or to cause harm. The risk of data loss caused by synchronization problems is a significant concern, particularly in multi-cloud or hybrid environments where data is stored in various locations. In the event of synchronization failure, outdated or erroneous versions of data may overwrite

precise copies, resulting in integrity breaches. In addition to denial-of-service (dos) attacks and ransomware threats, the problem is worsened by the fact that attackers can encrypt, delete, or modify stored data, rendering it inaccessible or unusable [7].

Compliance and regulatory requirements, such as gdpr and hipaa, necessitate stringent integrity measures, but ensuring compliance in third-party cloud environments is intricate and demands continuous monitoring. Relying on third-party cloud providers introduces an additional layer of vulnerability, as organizations must place their trust in providers to implement strong security measures and prevent data integrity breaches. Unfortunately, certain cloud providers may not provide sufficient information about their security measures. The absence of effective auditing and monitoring systems makes it difficult to identify integrity breaches promptly, resulting in delayed response actions and amplified consequences of data corruption. Various blockchain-based solutions, cryptographic hashing, merkle tree verification, and third-party auditing mechanisms have been suggested to tackle data integrity issues, but they may not be practical for all organizations due to the associated computational and financial burdens. Furthermore, data redundancy techniques, such as erasure coding and replication, can improve data integrity but demand substantial storage resources [8]. To overcome these challenges, organizations must implement a multi-layered approach involving encryption, real-time integrity verification, continuous monitoring, and secure access controls. By addressing these issues proactively, businesses can ensure that their cloud-stored data remains trustworthy, accurate, and resistant to integrity breaches.

Numerous data integrity problems can make cloud service providers look bad and turn into users' worst nightmares. For instance, malicious activity can be used to purposefully or inadvertently alter data, exploit any flaw in the shared multi-tenant model to harm other users' data, cause data backup failures, cause data leaks, etc. [9]. Only secondary storage (non-volatile data) can use the probabilistic data integrity systems covered in this survey, which also clearly show tampering. This survey does not apply to volatile integrity schemes. The data owner outsourced by the data provider of the cloud storage (CSP), to which the data is outsourced, and the auditor that verifies the integrity of the data is three components of the data integrity scheme. The auditor may delegate this liability to a third -party auditor (TPA) or the data owner itself. The data integrity procedure consists of the following two phases:

- The preprocessing phase involves preprocessing the data and creating some extra metadata. The CSP is then closed for data processing and metadata.
- Verification phase: The auditor challenges the CSP that uses data and metadata to create evidence of holding and presenting an auditor. The auditor then verifies the evidence to ensure the maintenance of the data integrity. The data integrity solution makes it possible to quickly detect any corruption or loss of data, allowing proper recovery measures. We have developed thematic taxonomy to improve the understanding of data integrity schemes based on the types of data they process, the species of the metadata they use, their abilities, implementing primitives they use and scenarios of deployment.

The data integrity must be maintained in the cloud storage to ensure that the stored information is correct, consistent and protected from unwanted changes. During his life cycle, different types of data integrity mechanisms are implemented. The key types of data integrity in the cloud storage include physical, logical, entity, reference, domain and integrity of repository redundancy. Physical integrity ensures that data is protected from hardware failure, disk corruption or physical damage due to environmental factors such as energy disorders, overheating or natural disasters. Cloud providers deal with the physical integrity of data redundancy implementation, fault tolerances and recovery mechanisms across more data centers [10]. Logical integrity prevents data corruption due to software errors, human errors or cyber threats. It ensures that data relationships, formats and structures remain intact despite frequent access and cloud access. Cloud databases promote the integrity of entity by ensuring that each stored record has a unique identifier. Primary keys and unique restrictions prevent duplicity and ensure that data remain precisely referred to across distributed cloud storage systems. The reference integrity ensures that the relationships between the data stored in different locations of the cloud remain valid. Limiting foreign keys and the rules of the database integrity prevent simulated or non -conforming records, which ensures that the interconnected data sets maintain their consistency. Cloud databases enforce the integrity of the domain by limiting the types of values allowed in specific fields. Data validation rules, schema constraints, and format checks prevent incorrect or inconsistent data from being stored. CSPs may implement custom integrity rules based on business needs. For instance, in financial cloud storage, additional constraints might be enforced to prevent unauthorized transactions or incorrect financial reporting. The data transmission integrity ensuring that data remains unaltered during transmission between users, applications, and cloud servers is essential. Techniques such as end-to-end encryption, cryptographic hashing, digital signatures, and checksum verification are used to detect and prevent tampering or data loss during transit [11]. Cloud providers use data replication and erasure coding across multiple geographic locations to ensure that a copy of the data is always available in case of failures. This prevents data corruption due to storage media failures or cyberattacks.

The data integrity in cloud storage may occurred with some advanced techniques that protect against unauthorized modifications, accidental corruption, and cyber threats. The cryptographic hashing and checksums (e.g., SHA-256, SHA-3) generate unique hash values for stored data. If the data is modified, the hash value changes, enabling quick integrity verification. Control sums (eg CRC, MD5) are also used to detect data corruption during transmission and storage. Blockchain technology ensures storage of data resistant to manipulation by maintaining a fixed book of all transactions. Each data block is cryptographically interconnected, making it impossible to change records without detection. Decentralized storage solutions use blockchain to verify secure and transparent data verification [12]. The data is arranged

hierarchically in the Merle tree, with the hash of every child's node contained in each node. This structure minimizes computing costs while maintaining integrity by comparing only the required hash, allowing effective verification of massive data files. Search (POR) ensures that cloud providers do not change or remove data using cryptographic challenges and answers to demonstrate the availability and correctness of data, while data demonstrations (PDP) allow cloud users to confirm that their data is intact without removal. Calculations can be done on encrypted data without decrypting it due to homomorphic encryption. This ensures the integrity of data by providing cloud providers safely processing data and at the same time prevent unauthorized modifications. The evidence of zero knowledge (ZKP) allows the cloud provider to prove that the data remains intact without revealing its content. This technique is essential to verify the integrity of maintaining privacy in sensitive applications such as health and finance. Deleting coding (eg Reed-Solomon encoding) divides data into multiple frozen fragments, allowing reconstruction of lost or damaged data. More repository storage and distributed GEOSMIC backup increases tolerance and integrity of faults. The AI-powered anomaly detection for integrity monitoring able to detect anomalous access patterns, unauthorized changes, and suspicious transactions in real-time, helping prevent integrity violations before they escalate [13].

Enhancing data security by double-encryption using A.E.S. is the primary goal of the current research. We have used data from the virtual cloud that contains specific characteristics like CPU, memory, network traffic, number of instructions executed, power consumption, energy efficiency, execution time, types of tasks, priorities, and status. To detect anomalies and examine any suspicious patterns that can compromise the integrity of the data, we have implemented machine learning techniques. The following is the paper's contribution:

- To enhance the security of cloud data by using AES algorithm and provide double-encryption.
- To extract the top features by using recursive feature elimination.
- Finally, AI-powered anomaly detection is proposed for data integrity and to detect anomalous access patterns, unauthorized changes, and suspicious transactions in real-time, helping prevent integrity violations before they escalate.

The remaining paper can be categorized into the following sections: section II focuses on the latest research which has been conducted for the data integrity in cloud storage. Section III discussed proposed methodology with the experimental setup configuration. Section IV focused on the dataset description with number of features and selected features of virtual cloud dataset. Section V demonstrated the results with discussion of results. Section VII concludes the with future scope.

II. LITERATURE REVIEW

In the current section several researchers contributed towards the maintenance of data integrity in cloud computing. The homomorphic tags with RSA-based have been employed as verification metadata in the method provided by Ateniese et al. [14]. Unforgeable, the tags allowed for blockless verification. However, the technique only applies to data that is static or appended. When dynamic data operations are used, it is impractical to recalculate tags for every block before the change point whenever a block is added or removed from the file. The PDP plan offered public verifiability with less robust possession assurances and did not restrict the number of challenges. By permitting modification, deletion, and append operations, PDP's dynamic data operations constraints were largely addressed in [15] in 2008. Yet, this approach did not support insertion operations. Scalability and efficiency were achieved by symmetrical encryption of keys and cryptographic hash algorithms. CSP can cheat on the interference of responses, because the calls were not accidental in this way, causing it to be untrustworthy. The data owner was only allowed to ask the integrity of his data. Each technique can only use one copy of the data.

Scientists have published a plan that expanded the PDP access application to multiple file replicas [16]. If a replica is corrupted or deleted, this scheme could create new replicas on demand. It used masking by using PRFs (Pseudo Random Functions) and can withstand storage server collusion assaults. Although MR-PDP cannot be extended to a multi-cloud scenario, its use is restricted to a single cloud with numerous storage servers. All PDP methods were able to identify instances of corruption or server misconduct, but they were unable to restore deleted data once they were discovered. One characteristic that the proof of retrievability (POR) schemes introduced to the PDP schemes is data recovery. For static data, the authors [17] proposed two POR methods. 1) Publicly verifiable BLS Signatures for brief questions and answers. For lengthy queries and brief responses, pseudorandom functions are used; they only provide private verifiability. Unforgeability, unbounded queries, and blockless verification were further features; yet, these systems were still unable to stop data leaks. The authors [18] fully supported the dynamic data activities (inserting, updating, erase and editing) while maintaining the basic properties of verification, non -permeability and unlimited search.

Using a verified skip list based on the order, this system supported dynamic data processing. When the researcher [19] introduced hailstones that focused on high availability using redundancy both inside and between servers and data recovery, the existing properties of PDP schemes have been maintained. Code of correction errors (IP -CC), mixed by PRFS, ECC and universal hash (UHFS), was used for Macs. Yu et al. [20] They used RSA, bilinear pairing and trees formation Merle Hash to verify BLS -based brands to achieve data dynamics and public verification. A crucial prerequisite for public auditability, privacy protection was absent from this plan. The schemes that could be audited by the public were the only ones that lacked privacy-preserving features. A data integrity technique that combined privacy protection and public auditing was proposed by authors utilizing random masking in conjunction with a homomorphic authenticator.

Additionally, they developed the idea of batch auditing, which enables the execution of several auditing jobs at once. However, when it got unsuccessful auditing replies, it resulted in performance overheads. To create a message authentication code (MAC), the authors used PRFs and universal hashing. Its lower storage overhead, despite its higher communication overhead, was one of its noteworthy advantages. Nevertheless, it was limited to a single cloud and only offered private verifiability. Against a security point of view, the plan was safe against pollution and response attacks. However, because the network-based method stops that a small part of the file is read without the reconstruction of the entire file, it is not suitable for online storage systems where reading operation is largely performed. However, they could be used for backup and archival systems. Improved the ability of PDP schemes by providing resistance. However, he still lacked dynamic operations and public verification. The system used codes fixing errors (FEC) to achieve data recovery. The system designed by authors has been implemented using RSA-based algorithms and the use of hash-1 and Sha-512), pseudo-random functions and pseudo-flying permutations. Both the storage of data and the use of bandwidth are efficient with this technique. However, despite supporting public verifiability and privacy protection, the technique is only applicable to static data. By employing hashes and skip lists, it enhanced the dynamic PDP performance. However, because this technique must store secret information to conduct the hashes, it did not enable 1 public verifiability and suffered from an increase in client-side storage.

SLA verification was added to PDP by the authors in order to keep the data within certain geographic constraints. The authors have used MACs for both network delay measurement and evidence of possession in order to ascertain the geolocation of the data. However, there is no guarantee that more copies will be instantiated outside of the specified region. The method only works with static data, lacks public verification and fails if the proxy server is introduced between the client and the server. Authors were the first to propose justice in data integrity systems, which is a defense against rogue clients who blame honorary CSP for damaging its data. This approach used a verified data structure (range of 2-3 trees), incremental signature technique and ECC; However, the ECC resulted in poor performance. This approach allowed dynamic data operations, but did not support public verification. The authors used Reed-Solomon codes based on the Cauchy matrices, to ensure robustness, and to detect data corruption at the bytes. This approach lacks public verifiability and data recovery, even if it provides robustness for dynamic data operations. To improve the efficiency of calculation and storage, the authors' method associated with Lagrangian interpolation. As a result, it is compatible with built-in low source and mobile devices. By obtaining reduced exponentials, he was able to generate evidence of ownership regardless of the difficulty of the task. The plan is predicated on witness concealing and zero knowledge proof, which prevents a hostile verifier from speaking with the prover to discover the underlying witness. The Okamoto protocol, which makes use of pseudorandom generators and permutations, is the foundation of the signature system. Nevertheless, public verifiability and dynamic data operations are not supported. According to the authors' plan remote outsourced storage shared by several users can be verified by the public while maintaining user privacy (user data is protected from external auditors). To facilitate public auditing, homomorphic MACs are utilized for storage efficiency and group signatures for homomorphic authenticators. Identifiers of blocks are stored in an index hash table. Notwithstanding its ability to protect privacy, the technique cannot detect minor corruptions in sampling, and the cost of calculation rises as the number of sampled blocks increases. Many users can benefit from traceability without it impairing the verification process's efficiency. In this context, traceability means that the original user can identify the signer by tracking down a signature on a block.

MHT, pseudorandom functions, and hashing were used by the authors. Deduplication, or the identification and removal of undesired duplicate copies, is paired with proof of ownership (POW) and provable data possession (PDP). Additionally, the program promoted public audits, which was not the case with previous initiatives. The authors presented the PDP paradigm in the context of multi-cloud, introducing high security, high performance, and transparent verification. This method presented a cooperative PDP (CPDP) strategy based on hash index hierarchy (HIH) and homomorphic verifiable response (HVR) that utilizes bilinear pairings. All the features of data integrity schemes are supported, including data recovery, public auditing, unforgeability, unbounded searches, privacy protection, block less verification, and data dynamic operations. But it didn't offer sound knowledge, and it can't withstand attacks like tag forging and data leaks. Merkle Hash Trees (MHTs) were used by Authors used to minimize bandwidth overhead. Additional significant contributions of this work include support for dynamic data, reduced disk seeks, and independent metadata size. Because the number of blocks in a challenge is not fixed, this technique is unique. Using latency-based approaches, the authors integrated constraint-based data geolocation with data integrity verification in a multi-cloud setting.

With a manageable processing burden, researchers exploited algebraic signatures to benefit from minimal network bandwidth. The method provided protection against malicious tampering, despite its shortcomings, which included the inability to provide data recovery or public verifiability, cryptographically secure signatures, and dynamic data operations. By using erasure codes to hide the access pattern and the concept of oblivious RAM, the authors were able to avoid linking challenged blocks to actual storage. Data with redundant encoding can be effectively updated with this approach. However, it increased the cost of network connectivity. With a focus on secure computation, it employed commitment-based sampling (CBS) with designated verifier signatures, bilinear pairings, and Merkle hash trees. Although this technique enables public verifiability, it lacks the ability to do dynamic data operations and restore data. According to the authors' successful PDP method, multiple CSPs collaborate to store and handle client data. After that, they presented a cooperative PDP (CPDP) method based on hash index hierarchy (HIH) and homomorphic verifiable response (HVR) from the bilinear pairings. Their plan satisfied the security property of knowing soundness, they claimed.

III. PROPOSED METHODOLOGY

The user logs in through the recommended system with the cloud storage provider. When the data is uploaded, it is used to increase the security of double stains (ie the use of A.E.S.). The data is now stamped and transmitted between replicas servers. On servers, encrypted data are converted to binary format and then a parity bit is added to detect errors. A trusted third -party auditor will receive a copy of the codes in the existing system to confirm the data integrity, but there is a possibility that a third -party auditor could endanger this process. The proposed method extinguishes data on each server in the multi-server settings and generates a different code for each data block. For each server, this code is generated and sent to a third -party auditor. Each server in multiple servers' configuration has regularly has hash data and third -party auditor compares the codes to confirm integrity. Harm hackers can damage or change data if they detect any inconsistencies in encoding.

The data is taken from the primary server when a renowned third -party auditor states that data is manipulated. The server coding method is used to fix damaged data if the data on the server is similarly tuned. In 1997, the US government organized a competition and the winner was an advanced encryption standard, or AES. Due to the small size of the keys and progress in computing power, the data encryption standard was insufficient. It works on blocks, which are collections of bits with fixed length because it is a block cipher. It takes an input block of a certain size, usually 128, and creates an appropriate output block of the same size. The second input needed for transformation is a secret key. The fact that AES has three different keys - 128 bits, 192 bits and 256 bits - must be understood. The secret key of any size is acceptable. The technique is designed to function with the keys of 128, 192 or 256. It works on one block of 128 bits at a time to create 128 bits of encrypted text. After the initial XOR (BitWise Addition Mod 2) with the original key (provided the length of the key 128) is ten laps. ByteSub, ShiftRow, MixColumn, and AddRoundKey are the four steps (layers) that make up these rounds, except for the final one. The MixColumn step is skipped in the tenth round. 16 bytes, each containing 8 bits, are created from the 128-bit input. A 4×4 matrix is used to arrange these. While the ByteSub and AddRoundKey stages only work with the bytes, the ShiftRow and MixColumn steps work with this matrix. Each element in this field is represented by a single byte (8 bits), and these bytes can be multiplied and added to create additional bytes. These are the key characteristics of this field.

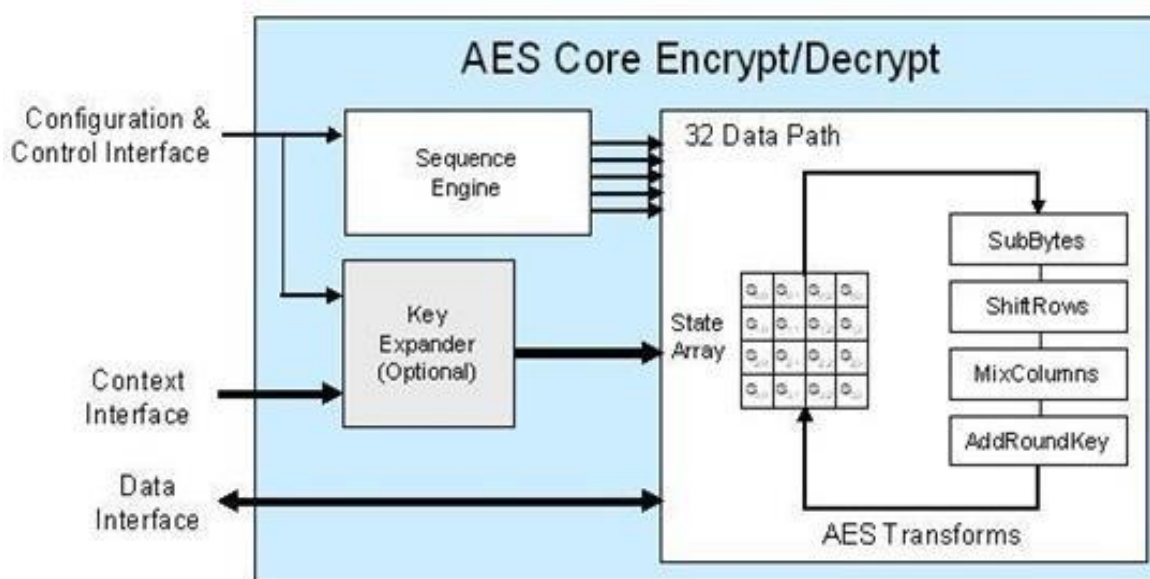


Fig 2. AES Encryption and Decryption.

FMSR codes have been using un-encoded data in the current system, which reduces security and makes it possible for a malevolent third-party auditor to attack. When software fails, data marts are lost, and client information is lost due to many reasons, such as file outages or network outages. The backup warehouse provides the copy to the data mart. Increases the availability of information. Data Mart S1 Loss of information P1, then can take S1 information from the backup warehouse and reconfigure it. If the Mart S2 and S3 data have lost this information, they can also get back information from the backup goods. MART data is a crash or the impact on the availability of information is also decreasing. The intended system also eliminates this disadvantage. Client requests can also be used to get data from the backup warehouse in the event that any data mart crashes or goes down. Data Mart S1 fails in the intended scenario and does not reply to user requests. In this instance, the P1 portion of the data is lost. The system's objective is to enable users to retrieve data from the backup warehouse. The availability of data marts has an impact on information security as well. The data is divided into many portions and stored in various data marts when there are a lot of them. The information in each data mart is extremely limited. An attacker can only take a small portion of the data if they manage to hack into any data mart. Cloud computing has drawbacks just like any other new technology. All users are generally impacted, however private individuals

or smaller enterprises may feel marginally more disadvantaged than larger companies due to factors including connecting peripherals, network connections, costs, lack of a hard drive, data ownership, and security. **Fig 2** shows AES Encryption and Decryption.

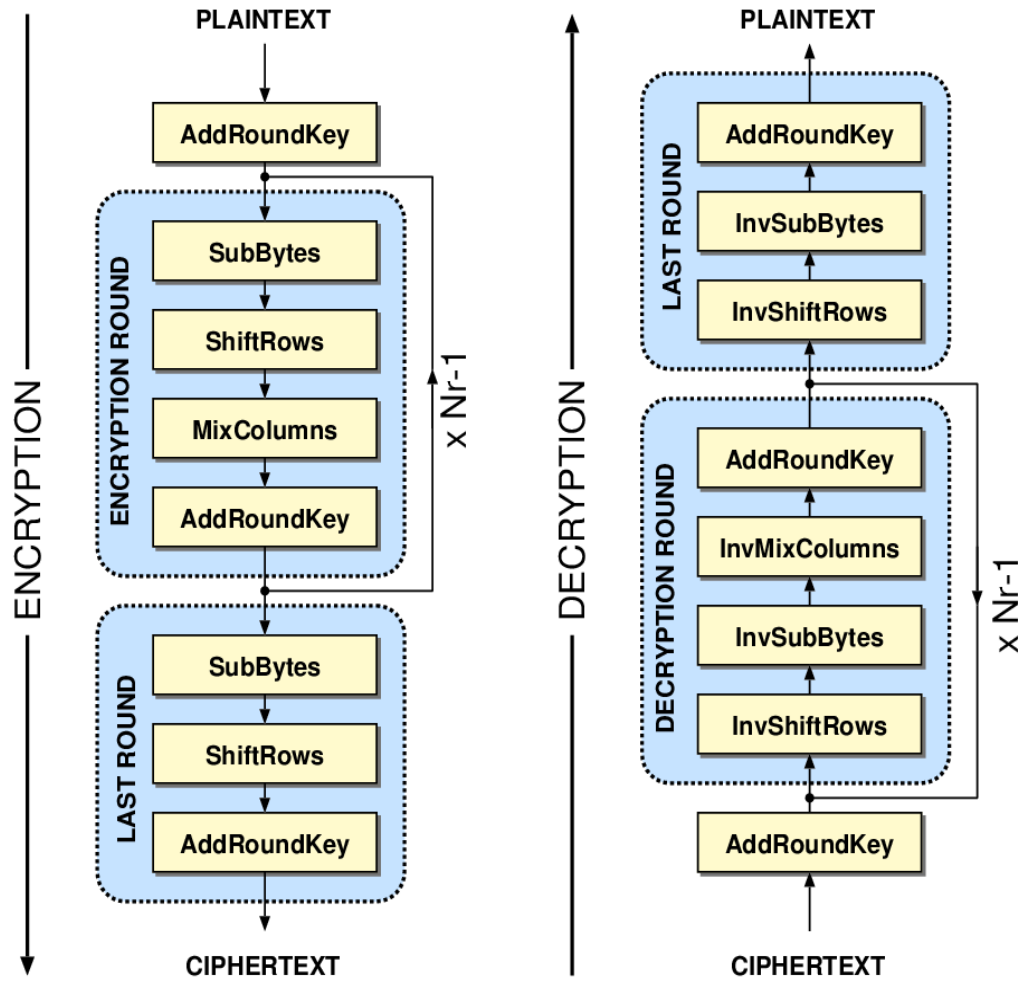


Fig 3. Encryption and Decryption Steps.

Fig 3 shows Encryption and Decryption Steps. After applying double encryption, machine learning algorithms are applied for the classification of suspicious and benign address. The linear classification is a baseline method for classifying the suspicious or anomalies in the cloud data. The linear classification is helpful when resources are scarce, it is unsuccessful when dealing with complicated datasets that exhibit non-linear patterns. A linear classifier predicts the y class based on different linear inputs x with bias b and weight w such as (equation 1):

$$y = w^T x + b \quad (1)$$

The classification results are depended on the decision rule in terms of equation 2 such as:

$$\hat{y} = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

An ensemble learning technique called Random Forest is used to identify intricate non-linear relationships in the data. A majority vote is produced by the ensemble of decision trees to categorize traffic as either harmful or benign. Compared to linear models, it uses greater computing complexity. Classification of random forest can be evaluated on the basis of equation 5, such as:

$$\hat{y}_{final} = Mode\{T_1(x), T_2(x), \dots, T_N(x)\} \quad (3)$$

The classifier aggregates different predictions of the tree decision -making such as $T_1, T_2, \dots, T_N$ and able to predict the final classification. Increasing the gradient is another file learning technique that is used to increase data integrity in

cloud computing by anomaly detection and prevention of unauthorized data modifications. It creates a strong predictive model using weak students to minimize errors in different epochs. The model is particularly effective in identifying fine formulas in cloud data sets and ensuring reliability and data security. This method seeks to minimize the loss function L using a weak student H on each epoch as defined in equation 4.

$$L(F_{m+1}(x)) = F_m(x) + \gamma_m h_m(x) \quad (4)$$

Where $F_m(x)$ is a file for iteration M , $h_m(x)$ indicates a Weak student to minimize residual error and γ_m represents the size of the step to evaluate $h_m(x)$. XGBOOST is another advanced file learning algorithm designed for high -performance, scalable and efficient detection of anomaly and verifying data integrity in cloud computing. It increases the traditional increase in gradient by incorporating regularization, parallel processing and optimization techniques for improvement and computing efficiency. It iteratively creates a set of decision -making trees and optimizes the loss function L and adds new trees to repair residual errors. The objective function consists of the loss and term of regularization (equation 5):

$$O = \sum_{i=1}^n L(y_i, \hat{y}_i) + \sum_{i=1}^m \varphi(T_k) \quad (5)$$

Where y_i indicates the actual value, $\hat{y}_i$ is the estimated value and $\varphi(T_k)$ is the term control to check the complexity of the model. Naive Bayes is based on Bayes' sentence and monitors the likelihood access to the detection of anomalies in large data sets. The likelihood of becoming in Bayes' sentence can be evaluated as in equation 6.

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} \quad (6)$$

The equation indicates the posterior probability $P(C_k|X)$ that can be evaluated for the class C_k and input X , $P(C_k)$ is indicating the prior probability for class C_k , $P(X)$ is the probability for input data X , and $P(X|C_k)$ is the likelihood of observations.

IV. DATASET DESCRIPTION

Benchmark evaluation of data set of trusted hosts (Beth) is designed to assess the security, integrity and performance of virtualized environments and cloud computing infrastructures. This data file captures the basic metrics related to VM, network traffic, performance and integrity to assess the reliability and security of cloud -hosted applications. The Beth data set can be used to detect disruption, detect anomaly in cloud environments. Benchmarking of performance can also measure virtualized infrastructure in terms of CPU, memory and disk use. The data integrity verification using cryptographic hash functions, validation mechanisms with threat modeling and attack simulation to evaluate cloud security frameworks. The key features are described in **Table 1**. The last two columns added manually for the monitoring of suspicious information such as 'sus' that indicating suspicious activity like an external userId with a systemd process and 'evil' shows the presence of external malicious process which is not inherent to the system.

Table 1. Feature Description with Data Type

Feature	Data Type	Description
Timestamp	Datetime	shows the exact time at which system boots.
ProcessID	Int	indicates the unique identifier for the process spawning
ParentProcessID	Int	Process spawning logs with parent's integer label
UserID	Int	Login Integers
MountNamespace	Int	Process logs with set mounting restrictions.
ProcessName	String	Name of the process.
HostName	String	Host server name.
EventID	Int	Event ID (Unique identifier).
EventName	String	Events logs generated names.
ARGSName	Int	Args length.
Return Value	Int	Event logs.
Stack Address	Int	Process related memory values.
ARGS	Float	Process that accepts the arguments.
SUS	Categorical	Indicates the suspicious event (1 is suspicious, 0 is not).
EVIL	Categorical	Malicious Event (0 is Benign, 1 is malware).

To generate and evaluate the BETH dataset, a controlled cloud computing environment is set up with the following configuration such as AWS EC2, Microsoft Azure, or OpenStack for VM deployment. There are multiple VM instances running on Windows or Linux operating systems with low, medium, or high compute power and RAM size is 2GB, 8GB,

and 16GB. CloudSim / Apache JMeter is used to generate synthetic workloads that simulate real-world cloud operations. There are more than 100000 number of rows with 5 sec per VM logging interval and 15+ attributes. **Fig 4** shows the number of values with the distinct attributes of the dataset.

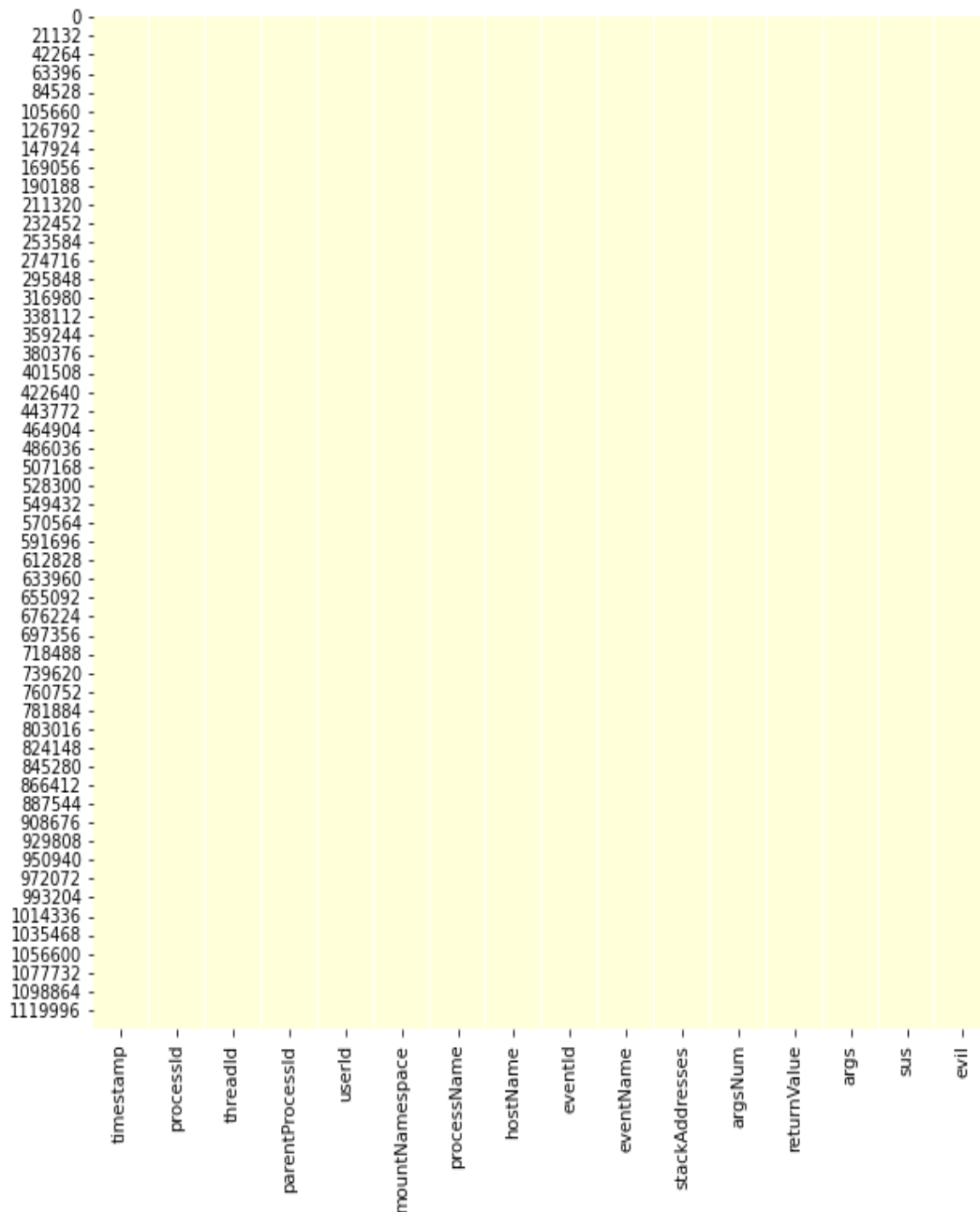


Fig 4. Dataset Attributes with Number of Values.

V. RESULTS AND DISCUSSION

This current section discusses the results in terms of data integrity, and anomaly detection for cloud computing. The **Fig 5** represents a heatmap visualization of the correlation matrix for various features in a dataset, which appears to be related to process monitoring or security analysis. A color gradient, with 1 denoting a perfect positive correlation, -1 denoting a perfect negative correlation, and values near 0 denoting little to no association, is used in the heatmap to show the correlation coefficients between various variables. The eventId and argsNum features exhibit a moderate correlation (0.63), indicating that the number of arguments used in a process may be dependent on the type of event occurring. Another significant observation is the negative correlation between mountNamespace and parentProcessId (-0.45), implying that processes with different mount namespaces tend to have different parent process behaviors. The process identifiers, including "processId" and "threadId," have a strong correlation of 1, which is expected since threads often belong to specific processes.

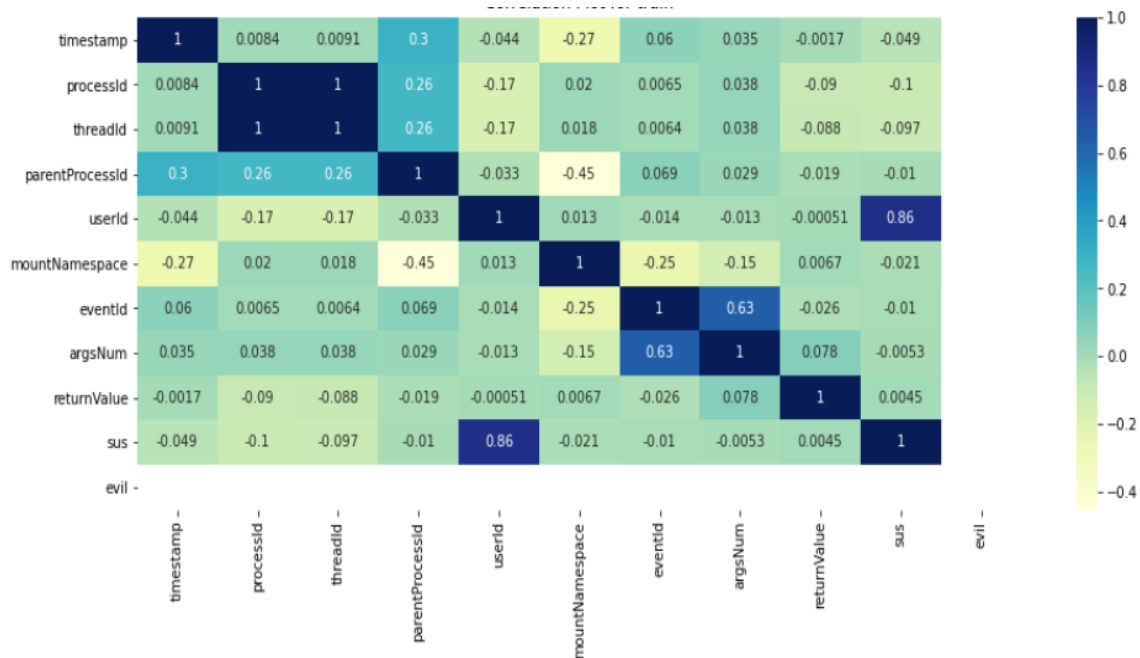


Fig 5. Dataset Attribute Correlation.

Top 10 Most Important Features:

	feature	importance
4	userId	0.272988
12	returnValue	0.184474
6	processName	0.150189
2	threadId	0.093406
13	args	0.091622
3	parentProcessId	0.090439
0	timestamp	0.045304
1	processId	0.036168
7	hostName	0.031924
9	eventName	0.002206

Fig 6. Top 10 Important Selected Features.

The **Fig 6** presents the top 10 most important features based on their significance in a ML model, used for process monitoring, anomaly detection, or security-related classification. The highest-ranked feature is user Id with an importance score of 0.272988, indicating that user identity plays a crucial role in distinguishing between normal and potentially malicious activities. The return Value follows with a score of 0.184474, indicating that the outcome of a process execution significantly impacts the classification or detection model. The process Name ranks third with 0.150189, reinforcing the importance of identifying specific processes in determining behavior. Furthermore, thread Id (0.093406) and args (0.091622) contribute meaningfully, implying that thread-specific behaviors and the number of arguments passed to functions are relevant in analyzing system events. The parent ProcessId (0.090439) shows that the hierarchical relationship between processes is an essential factor, potentially for tracking lineage and identifying suspicious process creation. The timestamp (0.045304) has a lower but notable influence, suggesting that time-based patterns are somewhat relevant. Finally, process Id (0.036168) and host Name (0.031924) indicate that tracking specific process executions and system-level identifiers contribute to detection accuracy. However, event Name has the lowest importance (0.002206), implying that individual event types alone may not significantly impact the model's decision-making. The **Fig 7** shows a bar chart representing the distribution of the timestamp across different data points and attributes. The x-axis corresponds to the index or unique identifiers of observations, while the y-axis represents the timestamp values, which could denote event occurrences or recorded time intervals in a system or network activity dataset. The chart exhibits a highly irregular pattern with significant spikes at certain points, indicating that specific timestamps have much higher occurrences compared to others. These peaks represent the attributes of those spikes that can flag anomalies or an unusual system event, security incidents, or process executions that are disproportionately concentrated around certain timeframes. The presence of numerous smaller bars across the graph indicates a varied distribution of timestamps, but the sharp spikes hint at anomalies or significant events occurring sporadically rather than uniformly over time. This type of pattern is often seen in system logs, where bursts of activity can occur due to scheduled tasks, attack attempts, or irregular user behavior. The relatively

stable baseline with frequent smaller values suggests that routine activities are consistently logged, while the outliers indicate moments of high-intensity logging. Analyzing such a distribution is critical in anomaly detection, performance monitoring, and security analysis. These spikes might correspond to attack attempts or abnormal process executions and can be analyzed for normal variations or potential threats requiring deeper investigation.

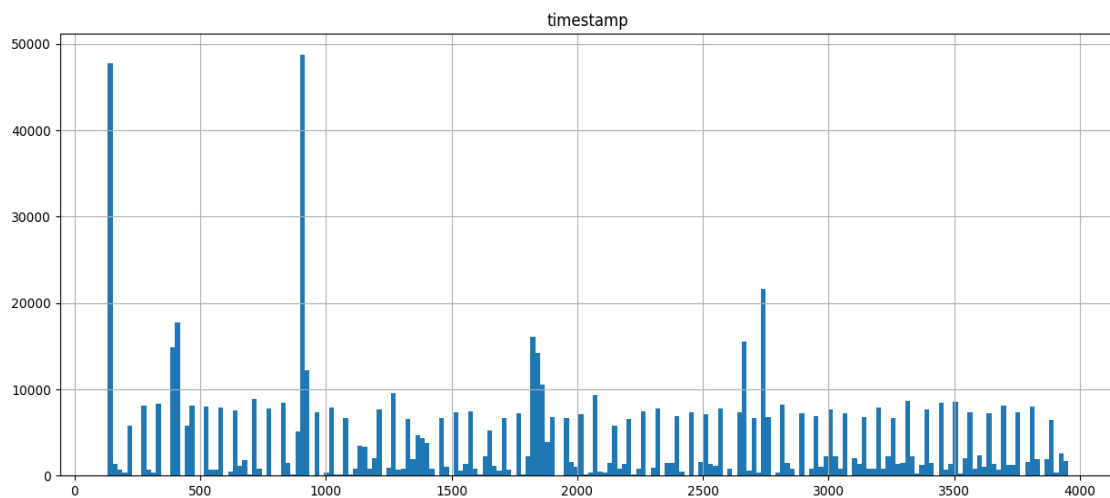


Fig 7. Bar Chart of Features with Timestamp.

When the value of *sus* is 1, it represents 1269 values corresponds to it that indicates the most of the events in the training data are benign while when its value is 0, it indicates 761875 number of values (**Fig 8**). Now, check the *userId* attribute for the suspicious field that indicate the top 3 rows are the suspicious events (*'sus' == 1*) where the value of *userId* is 1000 (**Fig 9**). That is the indicator of a suspicious event where there are no *userId* events of 1000 in the benign category.

<i>sus</i>	<i>userId</i>	Totals
1	0	237
1	100	36
1	1000	996
0	0	754670
0	100	1112
0	101	3398
0	102	1208
0	103	11
0	109	1476

Fig 8. Count Suspicious Data.

<i>sus</i>	<i>userId</i>	Totals
1	0	10163
1	109	45
1	1000	264
1	1001	160987
0	0	17332
0	100	10
0	101	82
0	103	84

Fig 9. Classification of Suspicious Data.

The userId cases of 1000 are still only seen in the benign cases, but now we can observe the huge number of userId 1001. These are the signs of anomaly and total 160987 number of values exists in the total cases. The **Fig 10** representing the bar plot with the distribution of userId over different timestamps. The x-axis denotes the timestamp values, while the y-axis represents different user IDs. The first cluster occurs at lower timestamp values (between 0 and 2000), where only a few user IDs show activity, with some appearing more frequently. A small, isolated green bar also appears around timestamp 7000, indicating a brief user presence. The second cluster in red, emerges between timestamps 13,000 and 17,000, showing significantly higher activity with consistent, densely packed occurrences of user IDs. The pattern suggests that a specific event or process triggered a surge in user activity during this time window, potentially due to automated system actions, coordinated access attempts, or a security incident such as an intrusion attempt. The userId values 1001 represents the evil in the test case.

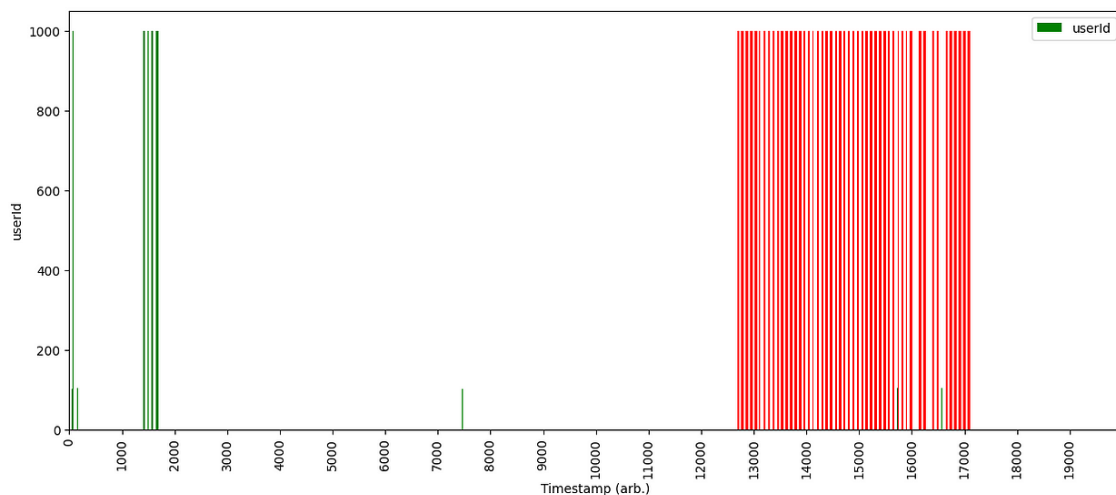


Fig 10. Distribution of Bar Plot for UserId and Timestamp.

The user Id attribute converted into a binary value, with userId values < 1000 being set to 0, and user Id values ≥ 1000 set to 1. The logic is to distinguish OS activity from user activity. The **Fig 11** presents the network traffic information, including the Source IP, Destination IP, and a count of occurrences for each connection. Additionally, the value of sus column indicates whether the connection is classified as suspicious (1) or non-suspicious (0).

sus	SourceIP	DestinationIP	
0	10.100.0.2	10.100.1.95	32
	10.100.1.95	10.100.0.2	32
	10.100.0.2	10.100.1.4	25
	10.100.1.4	10.100.0.2	25
	10.100.0.2	10.100.1.26	23
	10.100.1.26	10.100.0.2	23
	10.100.0.2	10.100.1.105	22
		10.100.1.186	22
	10.100.1.105	10.100.0.2	22
	10.100.1.186	10.100.0.2	22
	10.100.1.95	10.100.0.2	4
	10.100.0.2	10.100.1.95	4
1	10.100.1.105	10.100.0.2	2
	10.100.0.2	10.100.1.105	2
	129.250.206.86	10.100.1.95	1
	141.212.123.189	10.100.1.26	1
	141.212.123.193	10.100.1.186	1
	185.232.64.121	10.100.1.186	1
	192.35.168.27	10.100.1.26	1
	209.17.96.42	10.100.1.26	1
	209.17.97.122	10.100.1.186	1
	209.17.97.58	10.100.1.4	1
	74.120.14.31	10.100.1.4	1

Fig 11. Classification of Network Traffic.

The figure provides insight into communication patterns between different IP addresses and can be useful for security analysis, anomaly detection, or intrusion detection systems. In the non-suspicious (sus = 0) category, certain IP pairs appear frequently, such as 10.100.0.2 $\rightarrow$ 10.100.1.95 (32 times) and 10.100.0.2 $\rightarrow$ 10.100.1.4 (25 times). This suggests that these

are common or legitimate communications within the network, possibly representing regular system or user activity. Other frequently occurring connections, like 10.100.1.4 → 10.100.0.2 and 10.100.1.26 → 10.100.1.4, indicate a pattern of recurring communication between these devices. In the suspicious (sus = 1) category, connections occur significantly less frequently, with most appearing only once or twice. Some external IP addresses, such as 129.250.206.86 and 209.17.97.58 interact with the internal network, which may indicate potential intrusion attempts or anomalous access. The presence of multiple unique IP addresses with minimal activity suggests that these might be unauthorized or scanning attempts trying to probe the network. The contrast between high-frequency non-suspicious connections and low-frequency suspicious ones suggests that frequent internal communications follow expected patterns, whereas rare and externally sourced connections could warrant further investigation.

sus	DnsQuery	
0	ssm.us-east-2.amazonaws.com	224
	motd.ubuntu.com	8
	_http._tcp.security.ubuntu.com	4
	_http._tcp.us-east-2.ec2.archive.ubuntu.com	4
	security.ubuntu.com	4
	us-east-2.ec2.archive.ubuntu.com	4
1	_https._tcp.download.docker.com	4
	pool.hashvault.pro	4
	version.bind	3
	d2h670heeuigaw.cloudfront.net	2
	download.docker.com	2
	researchscan541.eecs.umich.edu	2
	213.1.168.192.in-addr.arpa	1
	4ac7e50e.openresolverproject.org	1
	c.afekv.com	1
	cpssc.gov	1

Fig 12. List of DNS Queries.

The IP combinations are associated with suspicious and non-suspicious events that can break-down the suspicious field via DNS queries. The **Fig 12** shows a list of DNS queries categorized under a column labeled sus, potentially indicating the level of suspicion or frequency of requests. The first column contains numerical values 0 and 1 representing different categories or levels of trust. The second column, DnsQuery, lists various domain names that were queried. The most frequently queried domain is ssm.us-east-2.amazonaws.com with 224 queries, suggesting it is a commonly accessed service, related to AWS systems manager. The second category 1 includes domains such as _https._tcp.download.docker.com, pool.hashvault.pro, and version.bind, which may suggest activities related to software updates, cryptocurrency mining, or DNS service queries. Some domains, such as d2h670heeuigaw.cloudfront.net and 4ac7e50e.openresolverproject.org, could indicate interactions with content CDN or open resolver. The presence of 213.1.168.192.in-addr.arpa suggests reverse DNS lookup activity.

After getting the DNS check, several machine learning algorithms are applied for measuring distinct performance parameters such as precision, recall, f1-score, and support. The **Fig 13(a)** displays the classification report of a random forest model, evaluating its performance based on precision, recall, f1-score, and support. The model predicts two classes: 0.0 (benign) and 1.0 (suspicious). The model accurately detects occurrences of innocuous class, as evidenced by the class 0.0 accuracy, recall, and f1-score of 1.00. Precision stays at 1.00 for class 1.0, but recall is little lower at 0.99, resulting in a f1-score of 0.99. This implies that some instances of class 1.0 are infrequently misclassified by the model. The number of genuine examples for each class is shown in the support column; there are 14,464 instances of class 1.0 and 55,781 instances of class 0.0. The model's total accuracy of 1.00 indicates almost flawless classification. The weighted average, which takes into account the distribution of classes, likewise reaches 1.00, as does the macro average, which gives equal weight to both classes. With 34958 supports, the gradient boosting approach displays a recall value of 0.99 (**Fig 13b**).

	precision	recall	f1-score	support		precision	recall	f1-score	support
0.0	1.00	1.00	1.00	55781	0	1.00	1.00	1.00	216080
1.0	1.00	0.99	0.99	14464	1	1.00	0.99	1.00	34958
accuracy			1.00	70245	accuracy			1.00	251038
macro avg	1.00	0.99	1.00	70245	macro avg	1.00	1.00	1.00	251038
weighted avg	1.00	1.00	1.00	70245	weighted avg	1.00	1.00	1.00	251038

Fig 13. (a) Results with Random Forest **(b)** Results with Gradient Boosting.

XGBoost uses boosting techniques, regularization, and tree pruning to handle large datasets and imbalanced classes, which improve generalization. However, in order to ensure robustness, it is important to validate the model using external test sets and cross-validation. The classification report of an XGBoost classifier is shown in **Fig 14(a)**, which evaluates its performance using key metrics like precision, recall, f1-score, and support. The model is classified between two classes: 0.0 and 1.0. For class 0.0, the accuracy, memories and score F1 will achieve a perfect score of 1.00, which means that the model correctly classifies all instances of this class without false positives or false negatives. For class 1.0, the accuracy remains at 1.00, but the download is slightly lower to 0.99, resulting in a score of F1 0.99. This suggests that even if the model is highly accurate, a small fraction of class 1.0 can be incorrectly classified. The support column shows that the data file contains instance of class class 0.0 and 34958, indicating an imbalance where class 0.0 is more frequent. The overall accuracy of the XGBoost model is 1.00, demonstrating near-perfect classification performance. The model's high recall and precision suggest it could be well-suited for applications requiring high reliability, such as fraud detection, medical diagnosis, or cybersecurity threat identification. Similarly, KNN algorithm is also utilized for result generation that indicates similar results as shows in XGBoost classifier except recall value is 1.0 for class 1.0 (**Fig 14b**).

	precision	recall	f1-score	support		precision	recall	f1-score	support
0	1.00	1.00	1.00	216080	0	1.00	1.00	1.00	216080
1	1.00	0.99	1.00	34958	1	1.00	1.00	1.00	34958
accuracy			1.00	251038	accuracy			1.00	251038
macro avg	1.00	1.00	1.00	251038	macro avg	1.00	1.00	1.00	251038
weighted avg	1.00	1.00	1.00	251038	weighted avg	1.00	1.00	1.00	251038

Fig 14. (a) Results with XGBoost (b) Results with KNN.

The **Fig 15(a)** represents the classification report of a Naïve Bayes model for Class 0 with 216,376 instances, achieves a perfect precision, recall, and f1-score of 1.00, indicating that the model correctly classifies all instances without false positives or false negatives. For class 1, which has 34,662 instances, the precision and recall are both 0.98, leading to an f1-score of 0.98. This suggests that while the model is highly accurate, a small fraction of class 1 instances is misclassified. The overall accuracy of the model is 0.99, signifying strong performance across both classes. The macro average, which provides an unweighted mean of the metrics, results in a precision, recall, and f1-score of 0.99, while the weighted average, which accounts for class distribution, also reaches 0.99. Given that Naïve Bayes assumes feature independence, its high accuracy suggests that the dataset aligns well with this assumption. The AdaBoost algorithm also utilized for accessing the performance that indicates the better results for class 0 with 216080 number of instances. While class 1 provides the precision, recall, and f1-score value 0.99 with 34958 number of instances (**Fig 15b**).

	precision	recall	f1-score	support		precision	recall	f1-score	support
0	1.00	1.00	1.00	216376	0	1.00	1.00	1.00	216080
1	0.98	0.98	0.98	34662	1	0.99	0.99	0.99	34958
accuracy			0.99	251038	accuracy			1.00	251038
macro avg	0.99	0.99	0.99	251038	macro avg	1.00	1.00	1.00	251038
weighted avg	0.99	0.99	0.99	251038	weighted avg	1.00	1.00	1.00	251038

Fig 15. (a) Results with Naïve Bayes (b) Results with AdaBoost.

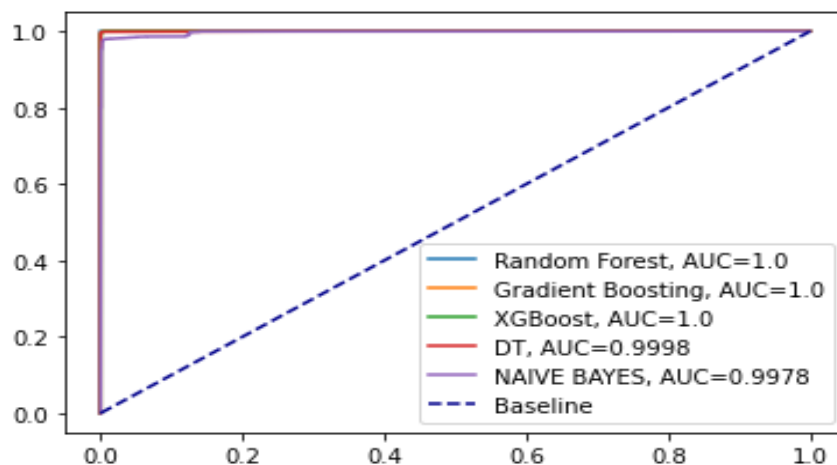


Fig 16. RoC for Distinct ML Model.

The **Fig 16** represents a ROC curve which is used to evaluate the performance of ML algorithm and plot the curve between TPR and FPR at various classification thresholds. The diagonal dashed blue line represents the baseline, which corresponds to random guessing with an AUC of 0.5. The curves for different classifiers, including Random Forest, Gradient Boosting, XGBoost, DT, and Naïve Bayes, demonstrate their classification performance. The AUC values indicate the model's ability to distinguish between classes, with values closer to 1.0 representing better performance. Random Forest, Gradient Boosting, and XGBoost achieve perfect classification with an AUC of 1.0, meaning they distinguish between positive and negative classes without error. The tree classifier is closely followed by AUC 0.9998, which shows almost perfect classification. Naive Bayes classifier has a slightly lower but still strong AUC 0.9978. The presence of multiple models with almost perfect AUC values suggests that the data file can be well structured or show strong separability between classes, allowing these models to achieve almost optimal performance.

VI. CONCLUSION

To maintain safety, reliability and trust, it is essential to ensure data integrity and detection of anomalies in cloud computing environments. ML algorithms have appeared as powerful tools to achieve these goals by analyzing huge data sets, identifying patterns and detection of deviations indicating harmful activities or corruption of data. The proposed system increases data security using a double encryption method (AES) and data are dissected and transmitted via replica servers. In the current system, a trusted third-party auditor is provided by a copy of data codes, there is a situation where it could become a threat of third food for this process. In the proposed system, the data on each server settings to set up multiple servers for each data block generated a unique code. This code generated for each server is given by a third-party auditor. Each server in multiple server settings has data at regular intervals and the third group auditor verifies the integrity by comparing codes. If the malevolent hacker notices any divergence in coding, it can damage or change the data. When the main server receives information from a trusted third -party auditor that the data is damaged, it gets data from the server. If the data on the server is also contaminated, the server coding method for defective data is activated. Finally, anomaly detection is being done by using distinct ML models and comparing them with performance metrics. The future scope of the current research is to maintain the data integrity by using blockchain and quantum cryptography.

CRedit Author Statement

The authors confirm contribution to the paper as follows:

Conceptualization: Aditi Joshi, Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; **Methodology:** Aditi Joshi and Rajendra Prasad Mahapatra; **Software:** Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; **Data Curation:** Aditi Joshi and Rajendra Prasad Mahapatra; **Writing- Original Draft Preparation:** Aditi Joshi, Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; **Visualization:** Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; **Investigation:** Aditi Joshi and Rajendra Prasad Mahapatra; **Supervision:** Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; **Validation:** Aditi Joshi and Rajendra Prasad Mahapatra; **Writing- Reviewing and Editing:** Aditi Joshi, Rajendra Prasad Mahapatra and Ganesh Gopal Devarajan; All authors reviewed the results and approved the final version of the manuscript.

Data Availability

No data was used to support this study.

Conflicts of Interests

The author(s) declare(s) that they have no conflicts of interest.

Funding

No funding agency is associated with this research.

Competing Interests

There are no competing interests.

References

- [1]. P. Prajapati and P. Shah, "A Review on Secure Data Deduplication: Cloud Storage Security Issue," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3996–4007, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.021.
- [2]. W. Li, W. Susilo, C. Xia, L. Huang, F. Guo, and T. Wang, "Secure Data Integrity Check Based on Verified Public Key Encryption With Equality Test for Multi-Cloud Storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 6, pp. 5359–5373, Nov. 2024, doi: 10.1109/tdsc.2024.3375369.
- [3]. A. Eghmazi, M. Ataei, R. J. Landry, and G. Chevrete, "Enhancing IoT Data Security: Using the Blockchain to Boost Data Integrity and Privacy," *IoT*, vol. 5, no. 1, pp. 20–34, Jan. 2024, doi: 10.3390/iot5010002.
- [4]. A. S. Alenizi and K. A. Al-Karawi, "Internet of Things (IoT) Adoption: Challenges and Barriers," *Proceedings of Seventh International Congress on Information and Communication Technology*, pp. 217–229, Jul. 2022, doi: 10.1007/978-981-19-2394-4_20.
- [5]. J. Li and R. Chen, "Development of an Accounting Informatization Model Through Cloud Data Integrity Verification," *International Journal of Information Technologies and Systems Approach*, vol. 18, no. 1, pp. 1–17, Feb. 2025, doi: 10.4018/ijitsa.368560.
- [6]. S. Feng, L. Deng, Y. Gao, Y. Wu, and J. Wen, "Blockchain-based remote data integrity auditing scheme with deduplication mechanism," *Cluster Computing*, vol. 28, no. 1, Oct. 2024, doi: 10.1007/s10586-024-04800-0.

- [7]. Q. Zhang, D. Sui, J. Cui, C. Gu, and H. Zhong, “Efficient Integrity Auditing Mechanism With Secure Deduplication for Blockchain Storage,” *IEEE Transactions on Computers*, vol. 72, no. 8, pp. 2365–2376, Aug. 2023, doi: 10.1109/tc.2023.3248278.
- [8]. G. Ateniese et al., “Provable data possession at untrusted stores,” *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598–609, Oct. 2007, doi: 10.1145/1315245.1315318.
- [9]. G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” *Proceedings of the 4th international conference on Security and privacy in communication networks*, pp. 1–10, Sep. 2008, doi: 10.1145/1460877.1460889.
- [10]. J. Yu and W. Shen, “Secure cloud storage auditing with deduplication and efficient data transfer,” *Cluster Computing*, vol. 27, no. 2, pp. 2203–2215, Jun. 2023, doi: 10.1007/s10586-023-04072-0.
- [11]. L. Deng, B. Wang, T. Wang, S. Feng, and S. Li, “Certificateless Provable Data Possession Scheme With Provable Security in the Standard Model Suitable for Cloud Storage,” *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 3986–3998, Nov. 2023, doi: 10.1109/tsc.2023.3303185.
- [12]. G. Chen, R. Hao, and M. Yang, “Popularity-based multiple-replica cloud storage integrity auditing for big data,” *Future Generation Computer Systems*, vol. 163, p. 107534, Feb. 2025, doi: 10.1016/j.future.2024.107534.
- [13]. J. Cai, W. Shen, and J. Qin, “ESVFL: Efficient and secure verifiable federated learning with privacy-preserving,” *Information Fusion*, vol. 109, p. 102420, Sep. 2024, doi: 10.1016/j.inffus.2024.102420.
- [14]. J. Yu, W. Shen, and X. Zhang, “Cloud storage auditing and data sharing with data deduplication and private information protection for cloud-based EMR,” *Computers & Security*, vol. 144, p. 103932, Sep. 2024, doi: 10.1016/j.cose.2024.103932.
- [15]. A. Mallick and R. P. Barnwal, “A Scalable Framework for Multi-cloud IoT Data Synchronization,” *Proceedings of the 26th International Conference on Distributed Computing and Networking*, pp. 364–369, Jan. 2025, doi: 10.1145/3700838.3703665.
- [16]. Q. Tong, L. Yin, Y. Liu, and J. Xu, “Append-only Authenticated Data Sets based on RSA accumulators for transparent log system,” *Computer Standards & Interfaces*, vol. 93, p. 103978, Apr. 2025, doi: 10.1016/j.csi.2025.103978.
- [17]. F. Tang, Y. Li, Y. Zhang, W. Susilo, and B. Li, “Real-time privacy-preserved auditing for shared outsourced data,” *Computer Standards & Interfaces*, vol. 92, Elsevier BV, p. 103927, Mar. 2025, doi: 10.1016/j.csi.2024.103927.
- [18]. S. M. Ganesh, S. Priya, V. Ravi, and S. A. Alsubibany, “A privacy preserving batch audit scheme for IoT based cloud data storage,” *Peer-to-Peer Networking and Applications*, vol. 18, no. 2, Jan. 2025, doi: 10.1007/s12083-024-01890-w.
- [19]. S. Ness, V. Eswarakrishnan, H. Sridharan, V. Shinde, N. Venkata Prasad Janapareddy, and V. Dhanawat, “Anomaly Detection in Network Traffic Using Advanced Machine Learning Techniques,” *IEEE Access*, vol. 13, pp. 16133–16149, 2025, doi: 10.1109/access.2025.3526988.
- [20]. L. P. Rachamalla, A. Akkidasari, S. Madiga, H. Mittapalli, and R. Vangipuram, “CLOUD ATTACK DATASET,” *IEEE DataPort*, Nov. 30, 2021, doi: 10.21227/05EP-ZK84.

Publisher’s note: The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. The content is solely the responsibility of the authors and does not necessarily reflect the views of the publisher.