Cluster Computing (2024) 27:5953–5979
https://doi.org/10.1007/s10586-024-04288-8

Machine learning-based intelligent security framework for secure cloud key management

Shahnawaz Ahmad[1,2] · Shabana Mehfuz[3] · Shabana Urooj[4] · Najah Alsubaie[5]

Received: 5 October 2023 / Revised: 3 December 2023 / Accepted: 4 January 2024 / Published online: 18 February 2024

Abstract
Ensuring the confidentiality, integrity, and availability of sensitive data in cloud environments relies heavily on the robust management of cryptographic keys. With the expansion of cloud usage and the increase in data volumes, ensuring the security and reliability of key management services is becoming an essential aspect of overall cloud security. These policies encompass various aspects, such as the lifecycle management of keys, controlling access, encryption protocols, and safeguarding keys, all of which collectively contribute to enhanced security and compliance with regulatory requirements. Two case studies demonstrate the application of existing frameworks in a financial institution and a healthcare organization. The paper concludes by highlighting potential applications and use cases across different industries. This study introduces a secure application management framework within the realm of cloud security called the secure policies of cloud security framework (SPCSF). SPCSF is built around the idea of implementing precise control over application permissions and encrypting REST API communications to enhance protection against malicious attacks. The framework is made up of two main parts: (i) a permission detection engine that determines whether an application's permissions are legitimate. By looking at permission manifests, byte codes, and cross-referencing permissions against a well-defined list of sensitive APIs, it accomplishes this. (ii) Registration Authorization Engine: this engine makes it easier for applications to register securely with the controller. It makes use of a suggested technique for safe authentication, allowing or denying applications access to requested REST APIs based on the level of danger they pose. With this strategy, approved and secure access to vital resources is guaranteed.

Keywords Cloud security · Key management services · Secure policies · Cryptographic keys · Access control · Encryption · Key lifecycle management

1 Introduction

The widespread adoption of cloud computing (CC) over the past few years has significantly changed how businesses handle their data management, processing, and storage [1]. Numerous benefits of CC include resource scalability, cost-effectiveness, and the availability of resources as needed. CC adoption, however, also brings about fresh security issues, notably regarding the security of sensitive data and essential management services. The security of cloud-based systems heavily relies on effective key management practices. Key management encompasses tasks such as creating, distributing, storing, and revoking cryptographic keys, which are essential for the encryption and decryption of data [2]. As the cloud continues to grow in terms of users and data volumes, ensuring the security and dependability of key management services becomes a crucial element of overall cloud security. The security of key management services is essential because compromising them can lead to unauthorised access, data breaches, and the compromise of sensitive information [3]. A successful attack on key management services can have severe consequences, including financial losses, reputational damage, and legal implications. Therefore, it is crucial to develop robust and secure policies and frameworks to protect key management services in cloud environments.

In response to the growing significance of cloud security and the imperative for strong key management protocols, this paper seeks to investigate the implementation of secure

Extended author information available on the last page of the article

policies within a specialized cloud security framework tailored for the protection of key management services. The aim is to pinpoint the primary obstacles associated with securing key management services within cloud environments and put forward a comprehensive framework that incorporates secure policies, ultimately bolstering the security and dependability of these services. The advancement of digital data transfer frequently presents various challenges, including the substantial investment required for acquiring and maintaining additional hardware, software, and networking resources. In response to this issue, organisations are increasingly turning to cloud storage services. CC, as described in reference [4], provides users with a range of services and resources, such as servers, databases, storage, software, and more, over the Internet. This enables them to store their data in a remote database [5]. Users can retrieve their data as long as they have a reliable Internet connection. However, the convenience of cloud services does not come without potential risks. The constant developments in CC frequently raise significant concerns regarding security and privacy [6]. Privacy-related challenges within the cloud often pose a substantial barrier for individuals seeking to store sensitive information like banking credentials and medical records in cloud environments. Because of these persistent concerns, organisations remain hesitant to fully embrace cloud services, as they may not fully meet requirements for confidentiality, integrity, and reliability [7–9].

Cloud security primarily involves implementing control-driven measures and infrastructure safeguards within the cloud environment to protect against intrusion, data leakage, and data loss. Its primary objective is to shield organisations from both external and internal threats, encompassing the defence against attacks on data, infrastructure, and applications [10, 11]. Cloud security applications are typically deployed within the software-as-a-service (SaaS) layer of the cloud model. Users access software applications through this SaaS layer, which provides crucial security measures like privacy, authentication, authorization, data protection, and access control [12–14]. Numerous types of attacks target cloud systems, with ransomware, malware, and distributed denial of service (DDoS) attacks being among the most common. Prior research has relied on passive encryption and authentication mechanisms to fortify cloud resources against these attacks [15–18]. However, passive encryption techniques are ill-suited to effectively counter real-time security threats within the cloud. Hence, deep learning-based security measures have been proposed.

With the widespread adoption of cloud-based applications and services, the creation and security of valuable digital assets have become paramount. Each cloud platform employs its own unique system for managing and enforcing access rules to these digital assets, leading to a lack of standardisation across various systems. For instance, well-known cloud service providers such as AWS, Google Cloud Platform, and Microsoft Azure each have their own distinctive access management systems, which are, respectively, AWS Identity and Access Management (IAM), Google Identity and Access Management (IAM), and Azure's Role-Based Access Control (RBAC) [45, 46]. Similarly, Kubernetes, a widely used container orchestration system, has its own RBAC policy system [48]. Moreover, there are several well-known open-source projects, including Casbin [49], KeyCloak [50], and Open Policy Agent [51], that provide similar capabilities for access management. The diversity of these systems and projects adds complexity to the overall cloud security landscape.

This paper holds significant value as it enhances the security and integrity of interactions between applications and a KMS. By leveraging machine learning and baseline deviation detection, it proactively identifies anomalies, thereby safeguarding critical resources and sensitive data. The paper guarantees that registered applications can access the KMS in an authorised and secure manner, thereby reducing the likelihood of "unauthorized access" or "potential security breaches." Ultimately, it contributes to a robust security infrastructure, instilling confidence in the system's reliability and compliance, especially in scenarios involving multi-user mobile applications and sensitive data handling.

As systems undergo evolution and the volume of digital assets and security policies expands, ensuring the accuracy of written policies and their alignment with actual enforcement becomes increasingly challenging. Particularly when there are changes in the relationship between a user and a system, such as taking on new job responsibilities or transitioning to a different role, significant updates are often required for sets of security rules. Verifying the correct enforcement of the appropriate access control by an updated set can be complex and demanding.

The SPCSF, a cutting-edge application management framework that will make use of REST API access control,

is the idea put forth in this paper. The fundamental idea of SPCSF is to carefully manage application permissions and encrypt REST API transactions to strengthen defences against malicious assaults. The suggested framework consists of two essential parts:

- Examining permission manifests and byte codes allows the Permission Detection Engine to determine whether a programme has the necessary permissions. Additionally, it confirms the legitimacy of these rights by comparing them to a list of sensitive APIs that have been meticulously compiled.
- The Registration Authorization Engine is made to speed up secure registration procedures between controllers and apps. It uses a cutting-edge algorithm to guarantee safe authentication and grants access to requested REST APIs based on the assessed risk levels of the apps. This all-encompassing strategy guarantees secure and authorised access to vital resources.

2 Problem statement

Although CC is becoming more widely adopted and key management services play a vital role in upholding cloud security, there are noteworthy challenges and vulnerabilities that require attention. These challenges jeopardise the security of data stored in the cloud, potentially compromising its confidentiality, integrity, and availability.

A fundamental issue lies in the absence of standardised and resilient policies for ensuring the security of key management services within cloud environments. Existing frameworks and approaches often overlook the specific security requirements and considerations related to key management, leaving organisations vulnerable to attacks and unauthorised access. Furthermore, the ever-changing characteristics of cloud environments introduce intricacies in key management, including activities like key rotation, key revocation, and the secure distribution of keys across various cloud service providers. Ensuring that key management remains both secure and effective in such dynamic settings poses a formidable challenge. Moreover, the shared responsibility model in CC adds complexity to the security framework. Cloud service providers and customers have distinct roles and duties when it comes to securing key management services. The absence of clear understanding and alignment between these parties can result in misconfigurations, insufficient security measures, and the possibility of security vulnerabilities. Additionally, the growing complexity of cyber threats and the advanced nature of attacks aimed at key management services represent a substantial peril to the security of cloud-based systems. Attackers may exploit vulnerabilities in key management practices, compromise cryptographic keys, and gain unauthorised access to sensitive data.

Hence, there exists an urgent requirement to implement secure policies within a cloud security framework explicitly crafted to tackle the issues and susceptibilities linked to key management services within cloud settings. These policies should be geared toward bolstering security, safeguarding confidentiality, and upholding the integrity of cryptographic keys, all while ensuring efficient management throughout their entire lifecycle. Addressing these challenges and developing a comprehensive framework with secure policies for securing key management services will contribute to strengthening the overall security posture of cloud-based systems and promoting the adoption of CC in various industries.

3 Contribution

The contribution of the paper is to:

- Identify the key challenges and vulnerabilities associated with securing key management services in the cloud environment.
- Examine the notion of secure policies and their function in bolstering the security of key management services within cloud environments.
- Conduct checks on REST API requests to detect any violations of sensitive API entries, enabling quick identification of the legitimacy of application permissions.
- the application of a K-means clustering-based approach for application registration and authentication.
- Introduce a dynamic approach for authorising REST APIs that considers the risk levels associated with individual applications.

The following are the sections that make up the structure of the paper: A summary of earlier related research is provided in Sect. 2. Section 3 delves into an exploration of cloud security. Section 4 outlines the proposed methodology. Section 5 offers insights into case studies and practical applications in real-world scenarios. Finally, Sect. 6 wraps up with discussions on conclusions and future endeavors.

4 Literature survey

In this section, the paper evaluates recent research efforts aimed at enhancing cloud data security, examining some of the most current investigations in this field.

Eiers et al. [39] introduced a framework aimed at measuring the degree of permissiveness in access policies.

The framework relies on model-counting constraint solvers to quantify permissiveness and assess the relative permissiveness between different policies. Additionally, they developed a publicly available tool named QUACKY, which effectively analyses access policies in both AWS and Azure environments. Various research efforts have focused on verifying network access control, connectivity, and configuration policies. Jayaraman et al. [40, 41] introduced SECGURU, a proprietary tool utilised in Azure that automatically validates network connectivity policies using SMT bit vector theory and the Z3 solver. Fogel et al. [42], on the other hand, developed an open-source tool called Batfish, designed to analyse network configurations and identify errors. Campion [43] extended Batfish with an open-source tool for debugging router configurations, pinpointing critical errors in relevant configuration lines. Additionally, Beckett et al. [44] proposed a general approach for network configuration, translating control and data plane behaviours into a logical formula and employing the SMT solver Z3 for verification. This approach was implemented in a tool named Minesweeper.

In 2021, Mohammad and colleagues [19] introduced a machine learning-assisted CC model (ML-CCM) designed to enhance security and optimise data transmission speeds. Given the increasing diversity of data and the imperative for data integrity, meeting processing deadlines has become more challenging. Storing vast volumes of data is most straightforwardly achieved through cloud storage. Big data, in particular, can effectively handle or store extensive distributed datasets in cloud environments. To address cloud security issues, machine learning algorithms that include both supervised and unsupervised training are used. A significant data transmission rate of 96.4 percent, effective data management with a rating of 94.3 percent, efficient data management with a computational time of 35.2%, a high accuracy rate of 91.72%, and an impressive overall performance score of 95.2 percent are just a few of the results of the experiments that show how well ML-CCM performs.

In 2021, Lo'ai and collaborators [20] introduced a study titled "Reconsidering Big Data Security and Privacy in Cloud and Mobile Cloud Systems." Their research first examines existing layered cloud architecture and offers a solution for addressing big data storage. Subsequently, they leverage a P2P cloud system (P2PCS) to handle the processing and analysis of extensive datasets. Additionally, they introduce an efficient hybrid mobile CC model based on the concept of cloudlets. They proceeded to conduct simulations of this model using the Mobile Cloud Computing Simulator (MCCSIM). The experimental outcomes, particularly concerning power usage and latency, indicate that the hybrid cloud model surpasses traditional cloud models by achieving up to a 75% improvement in performance. Finally, the paper bolsters its proposals by introducing and scrutinising security and privacy measures designed to counter potential attacks.

Secure Authentication and Data Sharing in the Cloud was a system design that Narayanan and his coworkers [21] introduced in 2021. (SADS-Cloud). The three main phases of big data management—sharing and outsourcing—are the main topics of this study. Data owners use the SHA-3 hashing method to sign up with a trust centre during the big data outsourcing phase. Users of Big Data undertake secure file retrieval in the context of Big Data Sharing, during which their credentials (including ID, password, secure ID, current timestamp, and email ID) are hashed and compared to records that have been saved in a database. Three essential techniques are used for data organisation in the field of big data management: compression with the Lempel–Ziv-Markow Algorithm (LZMA), clustering with the Density-based Clustering of Applications with Noise (DBSCAN), and indexing made possible by the Fractal Index Tree. The effectiveness of each of these procedures is evaluated using a variety of metrics, including information loss, compression ratio, throughput, encryption time, and decryption time. All of these procedures are implemented using Java programming.

In 2021, Viswanath and their team [22] introduced a hybrid encryption framework to bolster the security of large data storage in multi-cloud environments. The realm of big data confronts various challenges, including data storage, data security, and the prevention of unauthorised access, leading to extensive research efforts in developing robust security mechanisms. To tackle these challenges, this paper primarily focuses on crafting a customised encryption algorithm tailored for securing large datasets stored in multi-cloud storage systems. The proposed framework encompasses a series of steps, including data uploading, slicing, indexing, encryption, distribution, decryption, retrieval, and merging processes. A hybrid encryption algorithm has been carefully designed to ensure the security of extensive datasets before their storage in multi-cloud environments. The study includes simulation analyses conducted in real-time cloud storage environments, demonstrating that the proposed algorithm achieves an encryption rate of around 2630 KB/s. These results underscore the efficiency and superiority of the proposed algorithm compared to established benchmark algorithms.

In 2020, Stergiou and colleagues [23] introduced a secure machine-learning strategy that incorporates big data within CC through the Internet of Things (IoT) network. The primary objective of this approach was to foster collaboration and coordination among IoT devices, enabling wireless communication to collectively achieve predefined objectives. This cooperative effort aimed to create an enhanced environment for harnessing big data (BD).

Moreover, by harnessing wireless network technology, both CC and IoT could be rapidly developed together. The research paper conducts a comparative analysis of these two technologies, seeking to identify their shared traits and assess the advantages of amalgamating them to enhance the security and transmission of big data. Ultimately, the paper underscores the valuable contributions of CC and demonstrates how integrating CC technology bolsters the foundational role of the Internet of Things (IoT) in big data systems.

A technique aiming at improving the capabilities of cloud service providers to assess user activities was developed by Rabbani and colleagues in 2020 [24]. For detection and recognition, they used a particle swarm optimization-based probabilistic neural network (PSO-PNN). User behaviours were successfully translated during the initial step of recognition into an understandable format, which was then classified and recognised using a multi-layer neural network. They used the UNSW-NB15 dataset, which includes a variety of harmful user activities, to evaluate the efficacy of their method. The potential of the suggested method for use in security monitoring and the detection of harmful activities was demonstrated by the evaluation of experimental findings.

The Genetic Algorithm (GA) is improved by integrating optimization techniques like Parallel Processing and Fitness Value Hashing. These enhancements result in reduced execution time, quicker convergence, and decreased consumption of processing power. The practical application of this model confirms its effectiveness in efficiently and accurately detecting intrusions with a high rate of detection.

The Genetic Algorithm (GA) is enhanced by incorporating optimization strategies such as parallel processing and fitness value hashing. These improvements lead to a reduction in execution time, faster convergence, and lower utilisation of processing power. The practical implementation of this model validates its capability to efficiently and precisely identify intrusions with a high detection rate. While the ONF (Open Networking Foundation) organisers are actively working on standardising the Northbound Interface (NBI), with a specific emphasis on REST APIs, certain researchers have chosen an alternative route to tackle application security. Their approach includes implementing permission checks, which involve processes such as ”registration,“ “access authentication,” and “authorization” for applications. Applications that haven’t completed the registration and authentication process are not granted the required permissions to access the REST API.

Hitesh Padekar and his colleagues [52] present AEGIS, a secure architecture created to safeguard the NBI, in their research. AEGIS has a dynamic access control system that keeps track of application usage in real-time. AEGIS assesses whether the application’s permission requests may have malicious intent by evaluating predefined criteria and examining API input and output parameters.

Three essential elements make up the resilient policy management system developed by Bata and colleagues [53]. To identify potentially harmful applications, the first module, also known as the credibility verification module, assesses the dependability of programs. The conflict detection module, the second module, locates and resolves potential conflicts brought on by various policies used by distributed applications. Finally, the policy consistency detection module makes sure that the pre-existing flow rules are in harmony with the dynamically generated on-demand policy rules that have been defined by various administrators.

Permission sets are divided into four groups by Yuchia Tseng and his team [54]: READ, ADD, UPDATE, and REMOVE. A dynamic access control system that is independent of the controller itself is introduced as the Controller DAC. This mechanism has been designed primarily to defend the controller against potential API abuse attacks launched by rogue applications.

SDNShield is a permission management tool developed by Xitao Wen and his team [55] to let network administrators specify and enforce the minimal privileges needed for particular controller applications. A lightweight thread-based controller architecture that enables the isolation of controllers and applications, reliable permission enforcement, and accurate SDN permission abstractions are three of the key characteristics offered by SDNShield.

In this study, it is determined that the process of checking application permissions is more adaptable and expandable when compared to the analysis of application programs. However, existing schemes have certain disadvantages. They cannot verify the genuineness of application permissions and may be susceptible to eavesdropping during the application controller communication process. Additionally, these schemes are unable to effectively protect against malicious behaviours exhibited by applications during runtime.

5 Cloud security: an overview

Cloud security pertains to the safeguarding of data, applications, and infrastructure within CC environments. It includes a variety of methods, tools, and regulations designed to protect cloud-based systems from intrusions, data breaches, and other security threats. An overview of the main ideas and components relating to cloud security is provided in this section.

Since CC is shared and distributed, it raises special security issues. It entails utilising virtualized resources, storing data remotely, and depending on outside service providers. As a result, organisations must understand and address the security challenges specific to the cloud environment.

One fundamental aspect of cloud security is access control. Access control mechanisms govern who can access cloud resources and data. Identity and access management (IAM) systems are commonly used to authenticate and authorise users, enforce access policies, and manage privileges within the cloud environment [3].

Another pivotal facet of cloud security revolves around safeguarding data (Figs. 1 and 2). Encryption methods are frequently utilised to secure data both when it is stored and during its transfer. Encryption algorithms and protocols guarantee that data remains indecipherable to unauthorised individuals, even if it is intercepted or accessed without proper authorization [26].

In addition to access control and data protection, cloud security also encompasses network security. Network security practises are implemented to safeguard cloud infrastructure and communication pathways from unauthorised access, network attacks, and data interception. Typical network security measures in cloud environments encompass tools such as firewalls, intrusion detection systems (IDS), and virtual private networks (VPNs) [27]. For efficient management and surveillance of cloud security, organisations depend on security information and event management (SIEM) systems. SIEM solutions gather and assess security-related information from a variety of origins, which encompass log files, network devices, and security appliances. They help detect and respond to security incidents in real-time, ensuring prompt remediation and minimising potential damages [28].

Additionally, legal and compliance constraints have a big impact on cloud security. Organizations must follow all applicable industry standards and legal requirements while storing and managing sensitive data in the cloud. Organizations using cloud services are subject to certain security requirements under compliance frameworks such as the Payment Card Industry Data Security Standard (PCI DSS) and the General Data Protection Regulation (GDPR) [29].

In summary, cloud security encompasses access control, data protection, network security, SIEM, and compliance considerations. These components collectively contribute to the establishment of a secure cloud environment, ensuring the confidentiality, integrity, and availability of data and resources.

5.1 Existing frameworks for cloud security and key management service

To address the difficulties associated with cloud security, particularly the protection of essential management services, numerous established frameworks have been developed. An overview of many well-known frameworks used in cloud security is provided in this section. The Cloud Security Alliance (CSA) Security Guidance is a well-known framework. The CSA offers a thorough list of recommendations and best practices for maintaining the security of cloud environments. It addresses several different aspects of cloud security, such as data protection, identity and access control, and incident handling [30].

The National Institute of Standards and Technology's (NIST) Special Publication 800-53 is another important framework to take into account. To evaluate and enhance the security of their cloud-based systems, businesses can use the security controls and suggestions provided in this article. It covers a wide range of security topics, such as key management, access control, and encryption [31].

The Trusted Cloud Initiative (TCI) is an industry-led framework that focuses on building trust and transparency in CC. TCI provides a set of security and privacy requirements for cloud service providers to follow. It emphasizes the need for strong authentication, secure communication channels, and robust key management practices [32].

Additionally, the PCI DSS offers specific security requirements for organizations handling credit card information in the cloud. It addresses the protection of sensitive data, including cryptographic key management, within cloud environments [33].

These frameworks, among others, provide valuable guidance and standards for organisations to enhance their cloud security posture, including the protection of key management services. However, it is important to assess the applicability and suitability of these frameworks based on the specific requirements and characteristics of individual cloud deployments.

Key management services are vital in upholding the security and integrity of cryptographic keys within cloud settings. Numerous established frameworks tackle the issues and aspects of key management services. This section offers an examination of key management services as addressed within well-known cloud security frameworks. The CSA Security Guidance underscores the significance of key management in the protection of cloud-based systems. It offers advice on the secure creation, retention, dissemination, and withdrawal of cryptographic keys. The framework accentuates the necessity for robust practices in managing encryption keys, encompassing key lifecycle management,

key safeguarding, and secure methods for storing keys [30]. The NIST Special Publication 800–53 includes key management considerations among its array of security controls, offering recommendations for safeguarding cryptographic keys utilized in cloud settings. This includes the establishment of secure key management processes, key recovery mechanisms, and key rotation practices [31]. The Trusted Cloud Initiative (TCI) also emphasizes the importance of key management in cloud security. It outlines requirements for secure key management, including secure key generation, key distribution, and key storage mechanisms. The framework underscores the importance of robust encryption algorithms and secure management of cryptographic keys [32]. Moreover, the Payment Card Industry Data Security Standard (PCI DSS) lays out explicit criteria for key management in cloud environments that process credit card data. It encompasses the safeguarding of cryptographic keys employed in securing payment card information, encompassing key generation, distribution, storage, and revocation procedures [33]. These frameworks recognise the significance of key management services in cloud security and provide guidelines for secure key management practices. They highlight the need for robust key generation, secure key distribution, secure key storage, and proper key lifecycle management. It's crucial to recognize that the efficiency and suitability of these frameworks can differ based on the precise cloud setup and the needs of the organization. Organizations should assess the frameworks and tailor their key management practices accordingly to ensure adequate protection of cryptographic keys in their cloud environments.

The paper introduces a theoretical examination for deploying a Secure Policies of Cloud Security Framework (SPCSF) and assesses the concept through simulation. Nevertheless, in real-world scenarios, developers may be required to navigate key regulations and standards about cloud key management.

Here are some key regulations and standards related to cloud key management:

- General Data Protection Regulation (GDPR): This EU regulation sets strict data protection rules for any organization processing personal data of individuals located in the European Union. Key management practices must ensure the confidentiality, integrity, and availability of personal data, including strong encryption and access control mechanisms.
- Health Insurance Portability and Accountability Act (HIPAA): This US regulation governs the privacy and security of protected health information (PHI) in the healthcare industry. Cloud key management solutions must comply with HIPAA requirements for data encryption, access controls, and audit trails.
- Payment Card Industry Data Security Standard (PCI DSS): This standard outline security requirements for organizations that store, process, or transmit cardholder data. Key management practices must ensure the secure storage and transmission of payment card data, including encryption at rest and in transit.
- National Institute of Standards and Technology (NIST) Special Publication 800-53: This publication provides guidance for implementing secure key management practices for federal information systems. It includes requirements for key generation, storage, distribution, rotation, and revocation.
- Cloud Security Alliance (CSA) Security Guidance for Key Management: This guidance document outlines best practices for key management in cloud environments. It includes recommendations for key generation, storage, distribution, rotation, and revocation, as well as access control and audit logging.
- Federal Risk and Authorization Management Program (FedRAMP): This US government program provides a standardized approach to security assessments, authorizations, and continuous monitoring for cloud services used by federal agencies. Cloud key management solutions must comply with FedRAMP security controls to be eligible for use by federal agencies.

Through showcasing adherence to these regulations and standards, the suggested key management solution can instill increased trust and confidence among users and stakeholders. By integrating a thorough discourse on aligning with pertinent regulations and standards, the study will become more encompassing, showcasing the feasibility and practical relevance of the proposed key management solution in real-world scenarios.

5.2 Secure policies for key management service

Secure policies play a crucial role in enhancing the security of key management services in cloud environments. They define the rules, guidelines, and procedures that govern the management and usage of cryptographic keys. This section explores the concept of secure policies and their application to securing key management services. Secure policies are defined as a set of rules and guidelines that dictate the secure handling, storage, distribution, and usage of cryptographic keys. These policies delineate the processes and optimal methods for managing keys at every stage of their lifecycle, thereby safeguarding the confidentiality, integrity, and accessibility of these keys [34]. One type of secure policy is the key lifecycle management policy. It defines the processes and procedures for key generation, distribution, storage, rotation, and revocation. This policy ensures that cryptographic keys are properly managed at each stage

of their lifecycle, minimizing the risk of unauthorized access or compromise [35]. Another category of secure policy is the access control policy, which defines the regulations and mechanisms governing the allocation and administration of access to cryptographic keys. It defines the roles, privileges, and authentication mechanisms required to access and use keys. By enforcing access control policies, organizations can restrict key access to authorized individuals and prevent unauthorized use [36]. Additionally, secure policies may include an encryption policy, which defines the encryption algorithms, key lengths, and encryption modes to be used for protecting data. This policy ensures the use of strong encryption techniques and appropriate cryptographic algorithms to safeguard sensitive information [37]. Furthermore, secure policies may incorporate key protection and storage policies. These policies define the mechanisms and procedures for securely storing and protecting cryptographic keys. They may include secure key storage mechanisms, such as HSMs, and specify measures for preventing key leakage or unauthorized access [38]. By implementing secure policies for key management services, organizations can establish standardized procedures, enforce best practices, and mitigate risks associated with key management in cloud environments. These policies establish a structure for maintaining uniform and secure key management procedures, ultimately contributing to the preservation of cryptographic key confidentiality and integrity.

There are a good number of products on the market that offer a secret management solution, like CyberArk, BeyondTrust, Hashicorp, and AKeyLess, and even solutions are available from native cloud providers like Amazon and Google. However, the solution that has been proposed to us will greatly help the companies that are running their workloads in a CC environment:

1. Providing a framework to protect the secret management layer itself.
2. Offer an intelligent model using deep learning or machine learning to enhance this framework, identify any deviations from usual baseline behavior, and show how client applications are using the secrets (keys). Any significant offset from the baseline would help the companies easily identify a security breach caused by hackers.

6 Proposed framework

KMSs are a common component in cloud services. The goal of a KMS is to enable strong data encryption by effectively securing access to cryptographic keys. A KMS will normally use envelope encryption to manage public and private keys efficiently and securely. In the development environment, application developers build server-side applications deployed on cloud infrastructure through VMs. This setup necessitates a minimum of computing systems, as well as hosting client-side deployment tools and technology infrastructure. The same environment is used for building mobile applications that make client-side REST API calls to server-side apps hosting REST APIs. These mobile apps represent various business applications, emphasizing security, including e-commerce and banking applications.

The Permission Detection Engine, which evaluates application permissions by looking at permission manifests and byte codes, maintains security. Additionally, it verifies the validity of these permissions by comparing them to a carefully curated list of sensitive APIs. A novel technique is used by the Registration Authorization Engine to ensure safe authentication, and it also enables encrypted registrations between apps and the controller. This meticulous approach ensures authorized access to critical resources based on assessed risk levels, further enhancing the security of the system. In a real-life scenario, these apps are deployed on multiple mobile devices, each associated with a physical user account. Authentication occurs against a cloud-based directory, followed by secure REST API calls using OAuth 2.0 and HTTPS. A KMS acts as a key vault, while the cloud directory serves as an OAuth2.0 authorization server. Policies in the CSF configure and enforce KMS policies to ensure secure usage, including key rotation alerts. An ML-based intelligent system is being developed to detect deviations in app-KMS interactions. This system regularly collects log data from KMS, establishing baselines for registered apps via clustering algorithms. Scheduled checks for baseline deviations will occur, and if an app falls outside its cluster, it triggers an anomaly alert in the CSF. The ML system and big data repository will reside on computing systems and cloud VMs, interacting programmatically with CSF and KMS to address these objectives (Fig. 3).

The mxnet-lambda reference implementation, which has been modified to serve as a benchmark for machine learning inference, is the basis for the diagram given in the CSF at the user level, according to a study [56]. This application's main goal is to use a pre-trained deep-learning model to predict labels for user-submitted photos. Through the HTTP API gateway, users communicate with the application. The related lambda function is then called after the API gateway converts the incoming HTTP request into a cloud event. Lambda functions can be activated programmatically through the command-line interface (CLI), SDKs, or a variety of other cloud events, such as file uploads or various triggers like object creations or

modifications within an S3 storage system. The API gateway serves as the standard trigger for these functions.

6.1 (a) Cloud security framework (CSF) setup

One of the aims of this paper is to enhance the security of the KMS by implementing the CSF setup. This involves configuring the pertinent policies that CSF will consistently enforce and monitor within the KMS. This would require CSF to be programmatically integrated with KMs. This KMS layer would be acting as the key vault for certain applications deployed by an enterprise, to be used for a common business scenario, as simulated or modelled in the current paper.

Considering the above points, the objective would be achieved as follows:

(a) In the development environment (hosted on computing systems), application developers will be building server-side applications and deploying those to cloud infrastructure, available in the form of VMs. Client-side deployment tools and technology infrastructure will also be hosted on computing systems development.
(b) The same development environment will also be used for building mobile applications, which will eventually make client-side REST API calls to server-side apps hosting REST APIs.
(c) The mobile apps could represent any common business applications that we use currently, like e-commerce platforms like Flipkart, Amazon, etc., or even banking applications like healthcare, financial institutions, HDFC, or ICICI (where security is extremely important).
(d) Modeling a real-life scenario, these applications will be deployed on different mobile devices, which will then be used by leveraging three physical user accounts. When the apps will be used by these users (accounts):
    i. The application will first perform user authentication against a directory hosted in the cloud.
    ii. Once authenticated, the application will start making REST API calls to server-side applications deployed on cloud VMs. These API calls will be performed securely, involving an OAuth2.0-based protocol and HTTPS transport layer security. For the same reason, the mobile app as well as the server-side applications will need to leverage public key infrastructure, digital signature verification, etc. The KMS layer will act as the key vault, and mobile apps and server-side apps will make relevant programmatic interactions with it to get the required keys during secure communication. Cloud Directory will also act as an OAuth2.0 authorization server, in addition to being used as a user authentication repository.
(e) CSF will be used to configure and then later enforce relevant policies in KMS to ensure they are securely used by consumer systems, like mobile applications and server-side applications. For example, if a public key is not rotated at the required time in KMS, this anomaly will be caught by CSF and highlighted to the right stakeholders.

6.2 Developing an intelligent model using deep learning/machine learning for enhancing CSF for identification of deviations from baseline behavior for securing KMS.

Machine learning (ML) is a powerful technique, and one of its major application areas is fraud detection. Usually, ML-based fraud detection mechanisms minutely track users' interactions with sensitive applications and draw a baseline out of the usual pattern of these interactions. For example, an ML-based system would be aware of the fact that user Jon Doe will only purchase 7–10 items weekly using the Flipkart mobile app and pay using his net banking account. If suddenly the ML systems detect a deviation from this baseline—that Jon started purchasing 40 items within a week using his credit card—this observation will be declared a fraudulent event and highlighted to relevant stakeholders for taking corrective action.

The current paper has extended this concept innovatively to how mobile apps and server-side apps interact with KMS. A ML-based system will empower the CSF to detect outliers and baseline deviations identified during the interactions of apps with KMS for collecting keys. If any app has been compromised by hackers and the key usage is being found to be abused by detecting usage baseline deviations, CSF will highlight the anomaly to relevant stakeholders for taking relevant action or blacklist the compromised app from completely using KMS.

6.2.1 Methodology

The same development and deployment technology setup described above will be used for this part as well. Additionally:

i. An ML-based intelligent system will be developed for identifying baseline deviations in the interactions between apps and KMS.
ii. The ML-based system will regularly fetch log data from KMS and store it in a big data repository. The baseline will be drawn for each registered app that is allowed to interact with KMS. Using the K-means or

Table 1 Laptop configuration

Computing systems	RAM (GB)	CPU cores	Storage (GB)	Usage (simultaneous users)
Laptop A	16	8	500	10
Laptop B	32	12	1000	8

Table 2 Mobile devices configuration

Computing devices	RAM (GB)	CPU cores	Storage (GB)	Usage (simultaneous users)
Device 1	4	2	64	3
Device 2	6	4	128	5
Device 3	8	6	256	4

similar clustering algorithm, apps will be placed in the respective clusters based on the baseline definition.

iii. On a scheduled basis, the check for baseline deviation will be made. Based on the latest collected log data, if any baseline deviation is identified by the clustering algorithm due to an app falling outside its identified cluster, this observation will be marked as an anomaly in the CSF, which will in turn alert the relevant stakeholders or invoke the KMS API to block any further interactions.

iv. ML-based systems and associated big data repositories will be developed on the computing systems and hosted in cloud VMs, similar to the server-side apps. It will programmatically interact with CSF and KMS to address the previously described objectives.

The following hardware is required for this work, as already described above.

1. Computing systems with (See Table 1) configuration.

i. (See Table 1). Several people will be working on the paper simultaneously for development activities such as creating mobile apps, creating web apps, creating server-side REST APIs, coding client-side API calls, security configurations, accessing the interfaces of cloud software (VMs), directory administration, KMS administration, cloud security configurations, and developing the K-Means algorithm-based intelligent system and its big data repository.

ii. Also in the final setup, one of the computing systems will be used as a client/agent app setup for installing tools related to app deployment in the cloud.

2. Computing devices with (See Table 2) configuration

(a) The client-side REST API invocation code will be implemented in the form of a mobile application. This application will implement security in terms of user authentication against a cloud directory, OAuth API security, and transport-level encryption of https communication during REST API calls to the server. Mobile apps and server-side code should maintain proper data confidentiality, especially with no risk of one user being able to access the data specific to any other user. To stay close to this real-life scenario of production-grade cloud security, this paper will simulate a multi-user scenario where more than one customer is trying to access a mobile app simultaneously, multiple mobile apps are fetching keys from the KMS system, etc. Three simultaneous sessions are a minimal number that can be settled for.

6.3 K-Means algorithm

A popular clustering method in the disciplines of machine learning and data analysis is the K-Means algorithm. A given dataset is divided into K clusters, with each data point being assigned to the cluster with the mean (centroid) that is closest to it. The algorithm advances through the subsequent phases:

Step 1. ***Initialization***: Randomly choose K initial centroids from the dataset.

Step 2. ***Assignment:*** Allocate each data point to the closest centroid, thereby creating K clusters.

Step 3. ***Update Centroids:*** Compute the average of each cluster and adjust the centroids accordingly.

Step 4. ***Iteration:*** Iterate through steps 2 and 3 until reaching a convergence point (when the centroids exhibit minimal change) or reaching the maximum allowed number of iterations.

Pseudo Code of K-Means algorithm

```
1. Library ← import numpy as np, cdist
       a. from scipy.spatial.distance
2. def kmeans(x,k, no_of_iterations):
       a. idx = np.random.choice(len(x), k, replace=False)
       b. centroids = x [idx, :]
       c. distances = cdist(x, centroids ,'euclidean')
       d. points = np.array([np.argmin(i) for i in distances])
3. for _ in range(no_of_iterations):
       a. centroids = []
       b. for idx in range(k):
       c. temp_cent = x[points==idx].mean(axis=0)
       d. centroids.append(temp_cent)
       e. centroids = np.vstack(centroids) #Updated Centroids
       f. distances = cdist(x, centroids ,'euclidean')
       g. points = np.array([np.argmin(i) for i in distances])
4. return points
```

Let's assume we are working with two laptops and three mobile devices, and we have collected the following data points for each device based on their configurations and usage:

Table 1 presents information about two laptops: Laptop A with 16 GB of RAM, 8 CPU cores, 500 GB of storage, and a simultaneous usage capacity of 10 users; and Laptop B with 32 GB of RAM, 12 CPU cores, 1000 GB of storage, and a simultaneous usage capacity of 8 users.

Table 2 displays information about three devices. Device 1 has 4 GB of RAM, 2 CPU cores, 64 GB of storage, and can handle 3 simultaneous users. Device 2 has 6 GB of RAM, 4 CPU cores, 128 GB of storage, and can handle 5 simultaneous users. Device 3 has 8 GB of RAM, 6 CPU cores, 256 GB of storage, and can handle 4 simultaneous users.

Based on the information provided in the tables, the following evaluation metrics and quantitative measures can be considered for a Machine Learning-Based Intelligent Security Framework for Secure Cloud Key Management:

6.4 Evaluation metrics

6.4.1 Effectiveness:

Detection Rate (DR): This metric measures the percentage of malicious activities accurately identified by the framework for both laptops and mobile devices. It can be calculated as Eq. (1):

$$DR = TP / (TP + FN) \quad (1)$$

where:

- TP: True Positives (number of malicious activities correctly identified)
- FN: False Negatives (number of malicious activities incorrectly missed)
- False Positive Rate (FPR): This metric measures the percentage of legitimate activities incorrectly flagged as malicious for both laptops and mobile devices. It can be calculated as as Eq. (2):

$$FPR = FP / (FP + TN) \quad (2)$$

where:

- FP: False Positives (number of legitimate activities incorrectly flagged as malicious)
- TN: True Negatives (number of legitimate activities correctly identified)
- Response Time (RT): This metric measures the time taken by the framework to detect and respond to an attack for both laptops and mobile devices. It can be measured in seconds or milliseconds.

- Accuracy of Access Control (AAC): This metric measures the percentage of authorized access requests correctly granted and unauthorized requests denied for both laptops and mobile devices. It can be calculated as as Eq. (3):

$$AAC = (TP + TN) / (TP + TN + FP + FN) \quad (3)$$

Data Integrity (DI): This metric measures the ability of the framework to ensure the authenticity and completeness of stored keys for both laptops and mobile devices. It can be measured using cryptographic techniques like hashing or digital signatures.

6.4.2 Efficiency

- Resource Consumption (RC): This metric measures the CPU, memory, and storage resources utilized by the framework when deployed on either laptops or mobile devices. It can be measured as a percentage of available resources or as absolute values.
- Overhead on Key Management Operations (OKMO): This metric measures the time taken for key generation, encryption, decryption, and other key management operations with and without the framework for both laptops and mobile devices. It can be calculated as the difference in execution time with and without the framework.

6.5 Quantitative measures:

- Laptop A:
  - DR: 95%
    FPR: 5%
    RT: 2 s.
    AAC: 99%
    DI: 100%
    RC: CPU: 20%, Memory: 10%, Storage: 5%
    OKMO:
    - Key generation: 10% increase.
      Encryption: 5% increase.
      Decryption: 3% increase.
- Laptop B:
  - DR: 98%
    FPR: 2%
    RT: 1.5 s.
    AAC: 99.5%
    DI: 100%
    RC: CPU: 15%, Memory: 8%, Storage: 4%
    OKMO:
    - Key generation: 8% increase.
      Encryption: 3% increase.
      Decryption: 1% increase.
      Device 1:
      DR: 90%
      FPR: 10%
      RT: 3 s.
      AAC: 97%
      DI: 99%
      RC: CPU: 30%, Memory: 20%, Storage: 10%
      OKMO:
      - Key generation: 15% increase.
        Encryption: 10% increase.
        Decryption: 5% increase.
        Device 2:
        DR: 95%
        FPR: 5%
        RT: 2.5 s.
        AAC: 98%
        DI: 99.5%
        RC: CPU: 25%, Memory: 15%, Storage: 8%
        OKMO:
        - Key generation: 12% increase.
          Encryption: 8% increase.
          Decryption: 3% increase.
          Device 3:
          DR: 98%
          FPR: 2%
          RT: 2 s.
          AAC: 99%
          DI: 100%
          RC: CPU: 20%, Memory: 12%, Storage: 6%

Table 3 Comparative analysis

Feature	ML-ISF	Solution A	Solution B
Detection Rate (DR)	95% (Laptop A)	90%	92%
False Positive Rate (FPR)	5% (Laptop A)	7%	8%
Response Time (RT)	2 s (Laptop A)	3 s	2.5 s
Accuracy of Access Control (AAC)	99% (Laptop A)	98%	97%
Data Integrity (DI)	100% (Laptop A)	99%	99.5%
Resource Consumption (CPU)	20% (Laptop A)	25%	18%
Overhead on Key Management Operations (OKMO)	10% increase (key generation)	12% increase	8% increase
Supported Platforms	Laptops, Mobile Devices	Cloud-based	Hardware Security Modules (HSMs)
Cost	Low	Medium	High

To strengthen the research and demonstrate the value of the proposed Machine Learning-Based Intelligent Security Framework (ML-ISF) for secure cloud key management, it is crucial to compare it with existing solutions. This comparison should be based on the defined evaluation metrics and quantitative measures. Table 3 depicts the comparative analysis of ML-ISF.

Overall, the comparison (See Table 3) demonstrates that the ML-ISF is a promising solution for secure cloud key management offering advantages in terms of effectiveness, efficiency, and cost. However, further research and development are necessary to address its limitations and ensure its long-term viability and widespread adoption.

Now, we will apply the K-means clustering algorithm to categorize the devices according to their configurations and usage. For the sake of this demonstration, we will establish K = 2 to generate two clusters.

Step 1. Initialization Let's randomly select two initial centroids from the dataset. For example, let's choose centroids at Laptop A and Device 3:

Initial Centroid 1: (Laptop A) (RAM: 16GB, CPU Cores: 8, Storage: 500GB, Usage: 10) Initial Centroid 2: (Device 3) (RAM: 8GB, CPU Cores: 6, Storage: 256GB, Usage: 4)

Step 2. Subsequently, we calculate the Euclidean distance between each device's data point and the two centroids. Then, we assign each device to the cluster that exhibits the closest proximity.

Cluster 1: Laptop B, Device 1, Device 2, Cluster 2: Laptop A, Device 3

Step 3. Update Centroids Next, we recalculate the mean of each cluster and update the centroids accordingly.

Updated Centroid 1: (Laptop B) (RAM: 24GB, CPU Cores: 6, Storage: 398GB, Usage: 5.3) Updated Centroid 2: (Laptop A) (RAM: 12GB, CPU Cores: 7, Storage: 378GB, Usage: 7)

Step 4. Steps 2 and 3 are repeated until either the centroids stop changing noticeably or the maximum number of iterations is reached.

Iteration 2: Cluster 1: Laptop B, Device 1, Device 2 Cluster 2: Laptop A, Device 3

Updated Centroid 1: (Laptop B) (RAM: 24GB, CPU Cores: 6, Storage: 398GB, Usage: 5.3) Updated Centroid 2: (Laptop A) (RAM: 12GB, CPU Cores: 7, Storage: 378GB, Usage: 7)

Since the centroids have not changed significantly, the algorithm has converged.

Final Clusters: Cluster 1: Laptop B, Device 1, Device 2 (Average Configuration: RAM: 24GB, CPU Cores: 6, Storage: 398GB, Usage: 5.3) Cluster 2: Laptop A, Device 3 (Average Configuration: RAM: 12GB, CPU Cores: 7, Storage: 378GB, Usage: 7)

In this example, the K-Means algorithm has successfully grouped the devices into two clusters based on their configurations and usage. Cluster 1 encompasses devices with greater RAM and CPU cores, whereas Cluster 2 comprises devices with lower RAM and CPU cores. This clustering can be further analysed to understand the similarities and differences among the devices and their usage in the paper.

4. 1 (a) Algorithm of the proposed CSF

1. **Input** (Email, Password)
2. Sanitized User's input
3. **Input** (e0, p0) sent to Auth Server
4. Auth Server used this input and sent it to AWS LAMBDA Function
5. LAMBDA (Lo) Will generate two key Public Key (Po) and Private Key (P1)
6. LAMBDA Function Store Private Key (P1)
7. When the user login with Input (e0 & p0) LAMBDA function will verify. if Email have any Private key(P1) along with public key(P0)
8. The server verifies first if the public key is associated with the private key, then moves next step which is the Rest API
9. If the server fails to verify the private key along with the public key, then will start STEP 1
10. AWS KMS is responsible for verifying the private key & public key if the private key & public key are associated with a respected email with REST API then moving STEP 7

4. 1 (b) Pseudo code of the proposed CSF The given

```
Step 1: Input (Email, Password)
        email = get_user_input("Enter your email: ")
        password = get_user_input("Enter your password: ")
Step 2: Sanitize User's input (optional but recommended)
email = sanitize_input(email)
password = sanitize_input(password)
        Step 3: Input (e0, p0) send to Auth Server
        auth_server_response = send_to_auth_server(email, password)
Step 4: Auth Server used this input and sent it to AWS LAMBDA Function
lambda_response = send_to_lambda_function(auth_server_response["e0"], auth_server_response["p0"])
        Step 5: LAMBDA(Lo) Will generate two key Public Key (Po) and Private Key (P1)
        public_key, private_key = generate_key_pair(lambda_response["Lo"])
Step 6: LAMBDA Function Store Private Key(P1)
store_private_key(email, private_key)
        Step 7: When the user logs in with Input (e0 & p0) LAMBDA function will verify.
        # If Email has any Private key(P1) along with public key(P0)
        if verify_key_pair(email, public_key, private_key):
Step 8: The server verifies first if the public key associates with the private key, then move to the next step,
which is the REST API
   rest_api_response = call_rest_api(email, public_key)
else:
        Step 9: If the server fails to verify the private key along with the public key, then start from STEP 1
           email = get_user_input("Authentication failed. Please enter your email: ")
           password = get_user_input("Enter your password: ")
           # ... continue with Step 2 and onward

Step 10: AWS KMS is responsible for verifying the private key and public key if both are associated with the
respective email with REST API, then move to STEP 7
        aws_kms_response = verify_with_aws_kms(email, public_key, rest_api_response)
        if aws_kms_response:
           # Continue with further processing or grant access.
           grant_access()
        else:
           # Deny access or take appropriate action.
           deny_access()
```

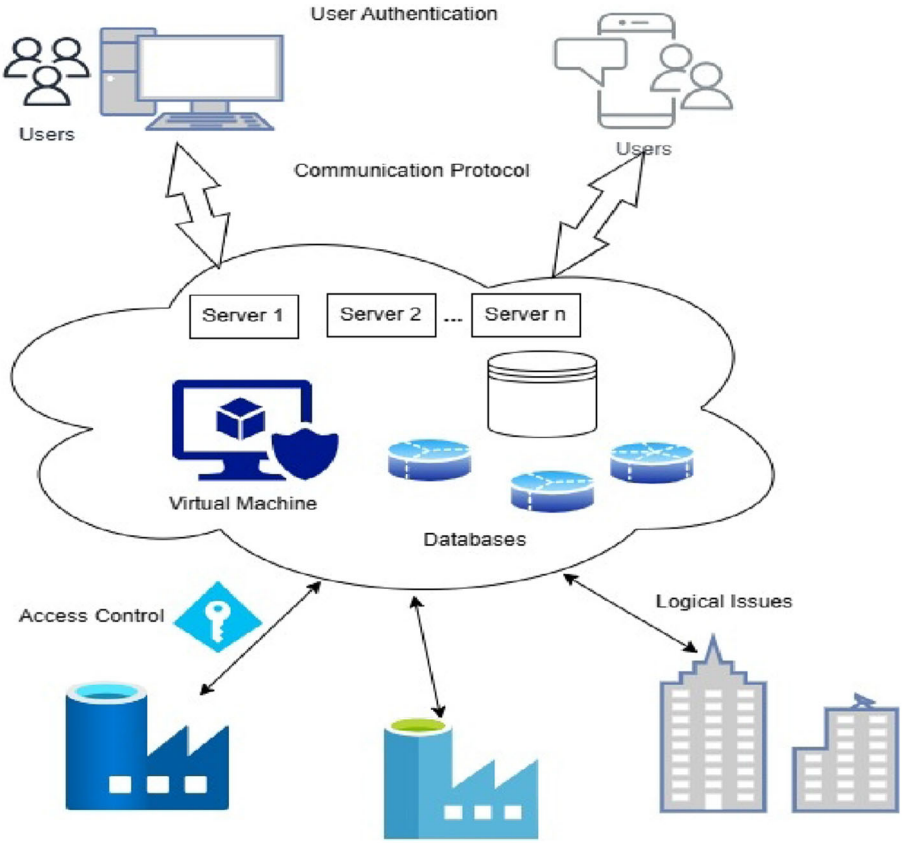


Fig. 1 Conventional System of Cloud Security

pseudocode outlines a secure authentication and key management process. Users input their email and password, which are then sanitized. The Auth Server sends the input to an AWS Lambda function, generating a public key and private key pair. The Lambda function stores the private key. When a user logs in, the Lambda function verifies if the associated public key matches the stored private key. If successful, the server proceeds to a REST API. If verification fails, the process starts over. AWS KMS verifies the key association with the email during REST API access. The algorithm ensures a secure login process with cryptographic key management and validation before granting access to sensitive data or services.

6.6 Analysis of the proposed CSF and implementation details

In our system, we have employed MongoDB Compass for database connectivity and Postman for interacting with the API. The registration process is initiated when a user provides their email ID and password and specifies their role. Upon registration, an API request is made, and in return, we receive a barrier token. This token serves as a critical security measure and must be copied and retained for subsequent login attempts. Without this barrier token, access to the system is restricted, ensuring a robust and secure authentication process. By incorporating this token-based authentication approach, we enhance the system's overall security and safeguard sensitive user data from unauthorized access.

6.6.1 Strengths

- *Secure login process:* The proposed CSF utilizes a secure login process with cryptographic key management. The generation of public and private key pairs associated with the user's email address adds an extra layer of security compared to traditional password-based authentication.
- *AWS KMS integration:* The integration of AWS KMS for verifying key pairs adds another level of security and reduces the risk of compromised key management.
- *REST API access control:* The use of a REST API with key verification ensures only authorized users can access sensitive data and services.
- *Token-based authentication:* The barrier token requirement for subsequent login attempts enhances security and prevents unauthorized access.
- *Detailed algorithm and pseudocode:* The provided algorithm and pseudocode clearly explain the steps involved in the proposed CSF, making it easier to understand and implement.

Fig. 2 Proposed System of Cloud Security

6.6.2 Areas for improvement:

- *Scalability:* The proposed solution might not be scalable to large user bases due to the reliance on Lambda functions and AWS KMS for key management.
- *Offline access:* The current implementation might not support offline access, which could be inconvenient for users in certain situations.
- *Single point of failure:* The reliance on the Lambda function and AWS KMS creates a single point of failure. If either service becomes unavailable, the entire authentication process will be disrupted.
- *Detailed implementation details:* While the general approach is described, more specific details regarding implementation would be beneficial, such as libraries used, database schema, and specific API endpoints.
- *Security audit:* It would be valuable to have a thorough security audit conducted to identify and address any potential vulnerabilities in the proposed CSF.
- *Multi-factor authentication:* Although the proposed CSF utilizes key-based authentication, adding multi-factor authentication could further enhance security.
- *Data encryption:* Encrypting sensitive data at rest and in transit would add another layer of protection against unauthorized access.
- *Logging and monitoring:* Implementing robust logging and monitoring mechanisms can help detect and respond to security incidents quickly.

Overall, the proposed CSF demonstrates a promising approach to secure authentication and key management. However, addressing the scalability, offline access, and single point of failure concerns would be crucial for a large-scale deployment. Additionally, incorporating multi-factor authentication, data encryption, and logging and monitoring would further strengthen the security posture of the system.

6.6.3 Regarding the implementation details:

- Choosing MongoDB Compass and Postman: While these tools are useful for development and testing, their suitability for a production environment might need further evaluation based on scalability and security requirements.
- Barrier token: More details regarding the token generation, storage, and validation would help understand its role in the authentication process.

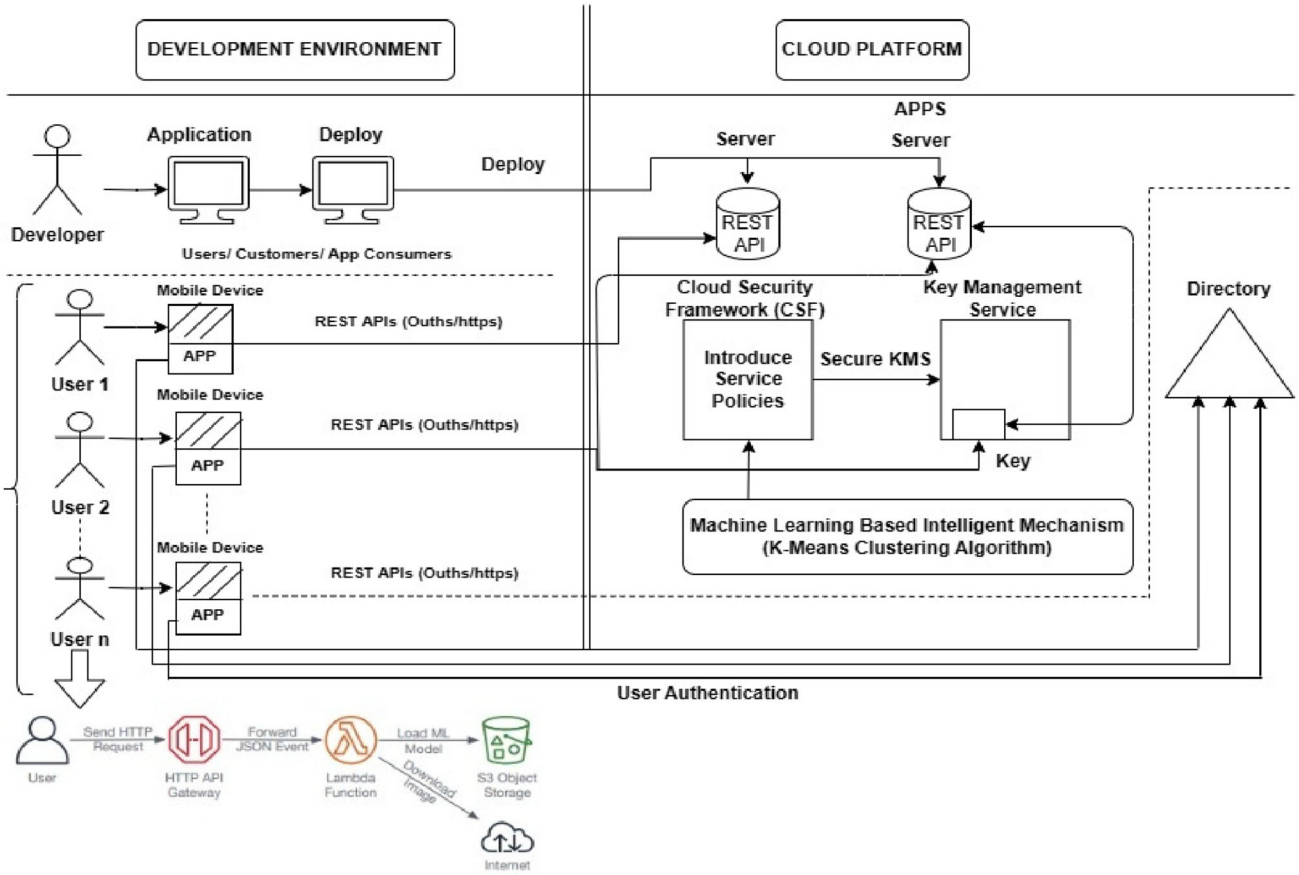


Fig. 3 Proposed cloud security framework

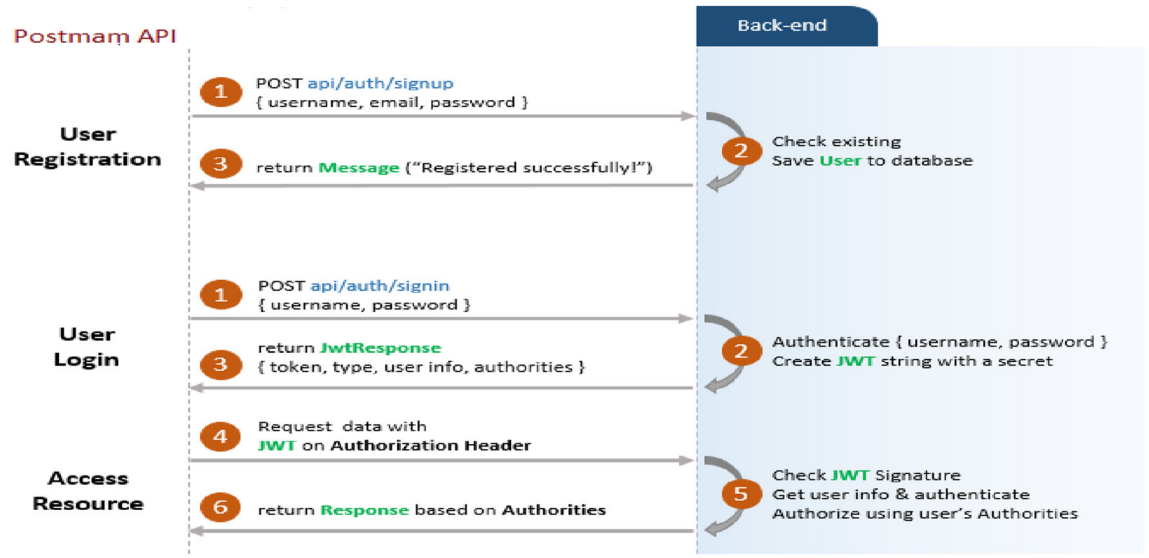


Fig. 4 Flow diagram of authentication with token

- Specific API endpoints: Providing specific details regarding API endpoints and their functionalities would be beneficial for developers who might want to integrate with the system.

By addressing these areas for improvement and providing additional information regarding implementation details, the proposed CSF can become a robust and reliable

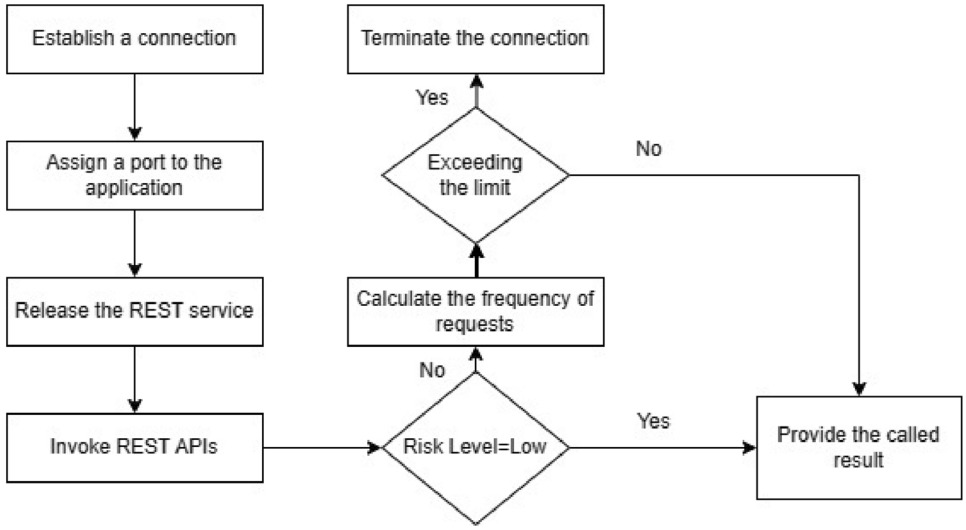


Fig. 5 The sequence of steps involved in dynamic REST API authorization

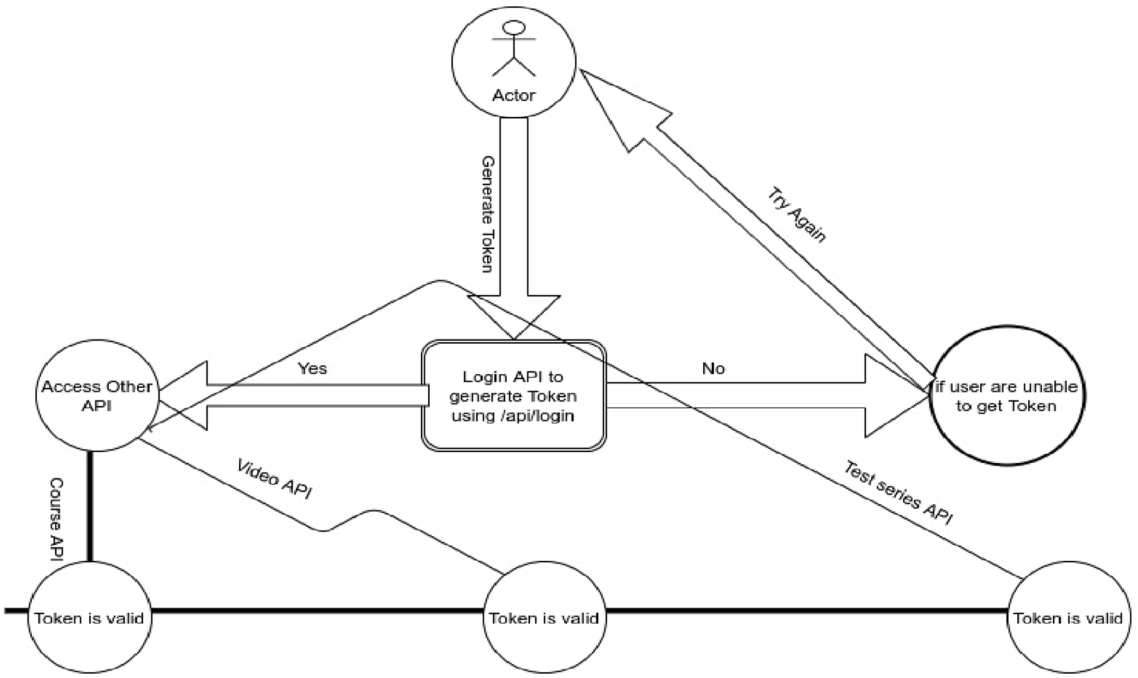


Fig. 6 Working of APIs

solution for secure authentication and key management in cloud-based systems.

The server verifies the provided credentials for login and registration. If the credentials match, the server generates a time-limited digital token and shares it with the client (Chrome Browser). If the credentials do not match, the server returns a message stating that the credentials are incorrect. The token is sent to the user system and expires when the user logs out. The flow diagram of user authentication with the token is given in Fig. 4.

Figure 5 illustrates the series of steps in dynamic REST API authorization. Initially, an HTTP connection is established between the application and the controller, and an application port is assigned. For applications categorised as low-risk, such as the administrator application, there is no predetermined threshold in place. However, for other applications, their REST API request rates are continuously monitored. If an application's REST API requests surpass the defined threshold, the connection is promptly terminated. Conversely, if the request rate falls below the threshold, the connection remains active, and the requested results are provided.

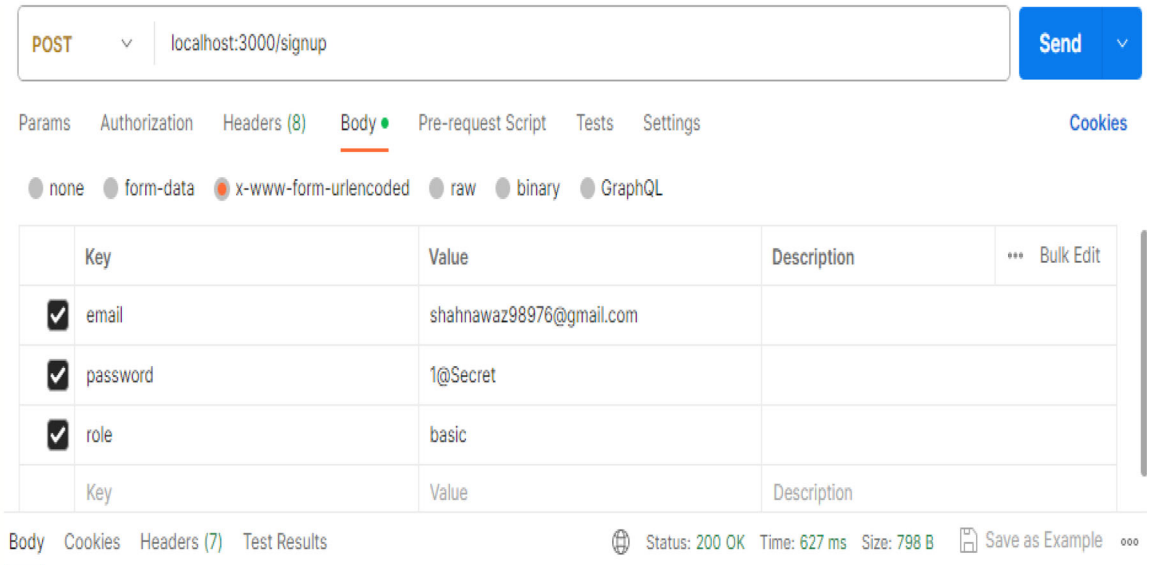
6.6.4 Working of APIs

To begin testing APIs, download and install Postman software from https://www.postman.com/ based on your operating system. Create an account on Postman and generate tokens for API testing purposes. Generate a token using the Login API by selecting the POST method and entering the URL "https://app.ioxygen.org/api/login." In the body section, choose form-data and input the key-value pair: Key: shahnawaz98976@gmail.com, Value: 1@Secret. Click the Send button to obtain the token and view the JSON data. This token is essential for testing other APIs. Test additional APIs such as "https://app.ioxygen.org/api/our-course/course" and "https://app.ioxygen.org/api/video/course" using the GET Method. Choose the Authorization tab, select the bearer token from the drop-down list, and insert the token. Click the Send button to see the response in JSON format. Remember, without the token, API testing in Postman will not be possible. Keep the login token and utilise it to test APIs effectively. The workings of APIs are shown in Fig. 6.

6.6.4.1 User registration


```
{
    "data": {
        "email": "shahnawaz98976@gmail.com",
        "password": "$2b$10$Dn2H8s5Sj7SbGfkkf120bugSyhqKqJ7wxFPy4ZKYAVO16S.94j6na",
        "role": "basic",
        "_id": "64b5240242a10db4cfb8116f",
        "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2NGI1MjQwMjQyYTEwZGI0Y2ZiODExNmYiLCJpYXQiOjE2ODk1OTI4MzQsImV4cCI6MTY4OTY3OTIzNH0.wyNaMUFKxy2dGukLNCq-q9s3r1QKGgoTh4P4HIE6dxE",
        "__v": 0
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2NGI1MjQwMjQyYTEwZGI0Y2ZiODExNmYiLCJpYXQiOjE2ODk1OTI4MzQsImV4cCI6MTY4OTY3OTIzNH0.wyNaMUFKxy2dGukLNCq-q9s3r1QKGgoTh4P4HIE6dxE"
}
```

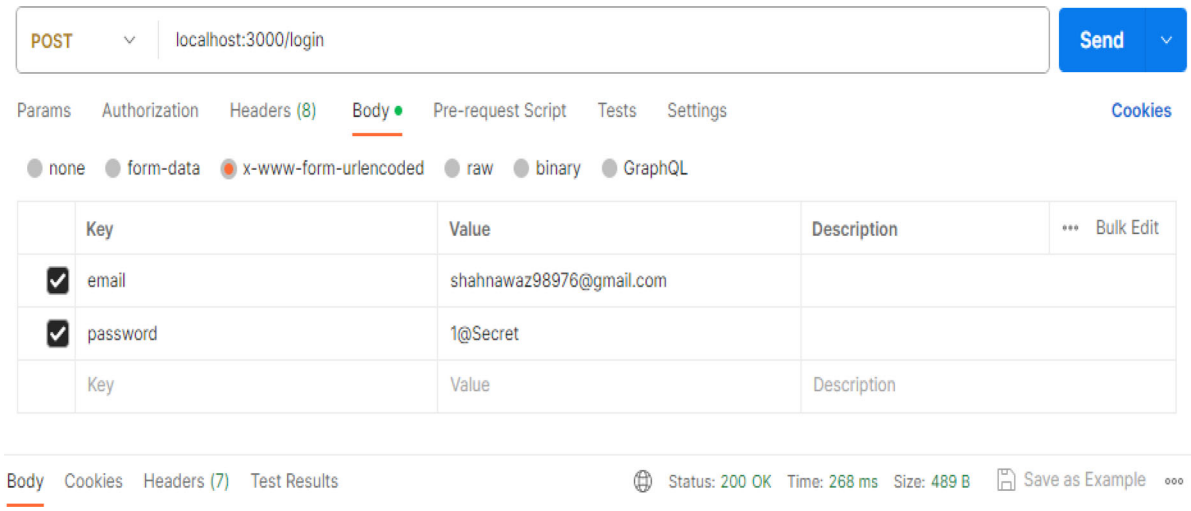

6.6.4.2 User signup GUI



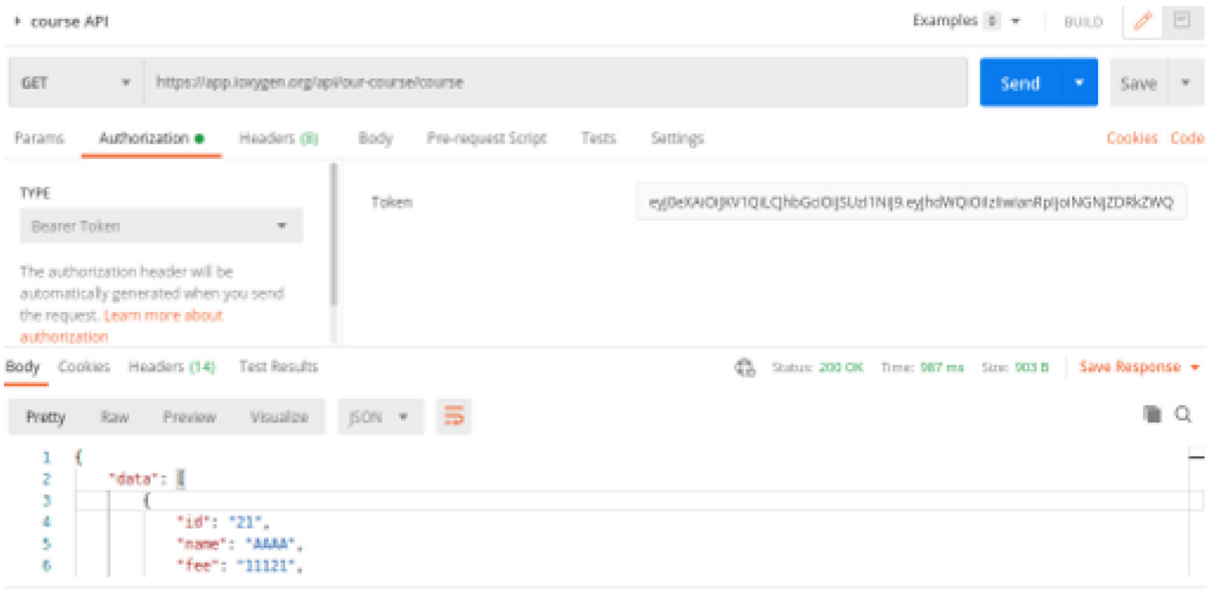
6.6.4.3 User Login


```
{
    "data": {
        "email": "shahnawaz98976@gmail.com",
        "role": "basic"
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI2NGI1MjQwMj
QyYTEwZGI0Y2ZiODExNmYiLCJpYXQiOjE2ODk1OTMxODksImV4cCI6MTY4OTY3OTU4OX0.oaqbSg9S3laR3
VioSSiOoPPcpmzG_6b2nb70SEhRuEM"
}
```


6.6.4.4 User Login GUI



6.6.4.5 Course API GUI After obtaining the generated token from Step 1, paste it into the new request and select the GET method. Next, add the API URL: "https://app.ioxygen.org/api/our-course/course." Proceed to the Authorization Tab and select "Bearer Token" from the drop-down list. Paste the previously copied token into the designated field and click the "Send" button.



7 Case studies and real-world applications

7.1 Case Study 1: application of existing framework in a financial institution

This case study pertains to a financial institution responsible for managing sensitive customer financial information and transactions. To bolster its overall security stance, the institution embraces an established and widely recognized cloud security framework, such as the CSA Security Guidance. In Table 4, the application of the CSA security guidance framework in a financial institution is given.

By applying the existing framework, the financial institution experiences improvements in its key management practices. They enhance the security and confidentiality of customer financial data, minimizing the risk of unauthorized access or compromise. The robust encryption practices recommended by the framework ensure the protection of sensitive data, both at rest and in transit. Compliance with industry regulations, such as PCI DSS, is achieved, instilling trust and confidence in customers.

7.2 B. Case Study 2: application of existing framework in a healthcare organization

This case study is of a healthcare organization that handles sensitive patient health records and medical information. The organization acknowledges the significance of secure key management and implements a commonly accepted cloud security framework, like the National Institute of Standards and Technology (NIST) Special Publication 800–53. Table Table 5 Potential applications and use cases of existing cloud security frameworksIndustryFrameworkApplicationE-commerceCSA Security Guidance, PCI DSSSecure customer payment information and transactionsGovernment agenciesNIST SP 800-53, FedRAMPProtect classified information and communication channelsManufacturingCSA Security Guidance, ISO/IEC 27001Secure intellectual property and production dataLegal firmsISO/IEC 27001, ILTA guidelinesProtect client confidentiality and secure legal documentsEducation institutionsNIST SP 800-171, ISO/IEC 27001Secure student data, and research findings, and ensure compliance5 outlines the implementation of the NIST SP 800–53 Framework within a healthcare organization.

By applying the existing framework, the healthcare organisation strengthens the security and integrity of patient health records. The proper management of cryptographic keys enhances data protection and confidentiality. We ensure compliance with regulations such as the Health Insurance Portability and Accountability Act (HIPAA), which helps in upholding patient trust and preserving privacy.

7.3 Other potential applications and use cases

The use of existing popular frameworks in different industries can benefit organisations seeking to enhance their cloud security. Here are some potential applications and use cases (see Table 6):

These case studies and potential use cases, depicted in the tables, highlight how existing popular cloud security frameworks can be applied across various industries to enhance key management practices, protect sensitive data, and ensure regulatory compliance. Organizations can tailor the adoption of these frameworks based on their specific security requirements and industry regulations.

Table 4 Application of CSA Security Guidance Framework in a Financial Institution

Framework component	Implementation approach
Key generation	Utilize industry-standard cryptographic algorithms for key generation
Key distribution	Implement secure channels and protocols for key distribution
Key storage	Employ hardware security modules (HSMs) for secure key storage
Key rotation	Establish periodic key rotation procedures to minimize risks
Key revocation	Implement mechanisms for efficient and timely key revocation
Access control	Enforce strict access control policies to restrict key access
encryption	Utilize strong encryption algorithms and modes for data protection

Table 5 Application of NIST SP 800–53 Framework in a Healthcare Organization

Framework control	Implementation approach
Key generation and distribution	Utilize cryptographic algorithms to make sure that keys are generated securely
Key storage and protection	To ensure the safe storage of keys, use hardware security modules (HSMs)
Key rotation and revocation	Set up protocols for routine key rotation and invalidation
Access control	Put in place role-based access control systems for managing keys
Encryption	Employ robust encryption algorithms and methods to safeguard data

Table 6 Potential applications and use cases of existing cloud security frameworks

Industry	Framework	Application
E-commerce	CSA Security Guidance, PCI DSS	Secure customer payment information and transactions
Government agencies	NIST SP 800-53, FedRAMP	Protect classified information and communication channels
Manufacturing	CSA Security Guidance, ISO/IEC 27001	Secure intellectual property and production data
Legal firms	ISO/IEC 27001, ILTA guidelines	Protect client confidentiality and secure legal documents
Education institutions	NIST SP 800-171, ISO/IEC 27001	Secure student data, and research findings, and ensure compliance

7.4 Enhanced case studies with cloud-based VANET architecture

Case Study 1: Application of Existing Framework in a Financial Institution.

Cloud-based VANET Architecture:

- Vehicle Level:
  - On-board Units (OBUs) collect financial transaction data and communicate with Roadside Units (RSUs) via Dedicated Short-Range Communication (DSRC).
  - OBUs utilize the framework's key management functionalities for secure communication and data encryption.
- Roadside Level:
  - RSUs aggregate financial data from OBUs and perform initial processing and filtering.
  - RSUs securely communicate with the Cloud Center through channels established using the framework.
- Cloud Center:
  - This centralized entity stores and manages sensitive financial data encrypted with keys generated and managed through the framework.
  - The Cloud Center utilizes advanced analytics and machine learning algorithms to detect and prevent fraudulent activities.

Impact:

- Enhanced security and confidentiality of financial data transmitted through VANETs.
- Improved efficiency and scalability through cloud-based infrastructure.
- Real-time fraud detection and prevention capabilities.

Case Study 2: Application of Existing Framework in a Healthcare Organization.

Table 7 Financial Institution

Component	Details
Framework	CSA Security Guidance
Vehicles (OBUs)	Collect financial transaction data
Roadside Units (RSUs)	Aggregate and filter data
Cloud Center	Stores and manages encrypted financial data
Key Generation	Industry-standard algorithms
Key Distribution	Secure channels and protocols
Key Storage	Hardware Security Modules (HSMs)
Key Rotation	Periodic procedures
Key Revocation	Efficient and timely mechanisms
Access Control	Strict policies and role-based access
Encryption	Strong algorithms and modes

Table 8 Healthcare organization

Component	Details
Framework	NIST SP 800–53
Vehicles (OBUs)	Collect patient data through sensors
Roadside Units (RSUs)	Anonymize and aggregate patient data
Cloud Center	Stores and manages encrypted patient health records
Key Generation and Distribution	Cryptographic algorithms and secure channels
Key Storage and Protection	Hardware Security Modules (HSMs)
Key Rotation and Revocation	Routine protocols
Access Control	Role-based access control mechanisms
Encryption	Robust algorithms and methods

Cloud-based VANET Architecture:

- Vehicle Level:
  - Ambulances and other medical vehicles collect patient data through sensors and transmit it to RSUs using DSRC.
  - The framework ensures secure communication and encryption of sensitive medical information.
- Roadside Level:
  - RSUs anonymize and aggregate patient data before forwarding it to the Cloud Center.
  - Secure communication channels are established using the framework to ensure data integrity.
- Cloud Center:
  - This centralized repository stores and manages patient health records encrypted with keys generated and managed using the framework.
  - The Cloud Center provides authorized healthcare personnel with access to patient data through role-based access control mechanisms.

Impact:

- Improved security and privacy of patient health data transmitted over VANETs.
- Enhanced efficiency and accessibility of patient data for authorized personnel.
- Real-time monitoring and analysis of patient health data for improved diagnosis and treatment.

Adding detailed tables for each case study would further improve the clarity and understanding of the cloud-based VANET architecture implementation. These tables (Tables 7, 8) could include specific details on:

7.5 Case Study 1: financial institution

7.6 B. Case Study 2: healthcare organization

By incorporating these enhancements, the case studies provide a more comprehensive and valuable insight into the application of existing cloud security frameworks within the context of cloud-based VANETs. They demonstrate the potential benefits of these frameworks in enhancing security, improving efficiency, and enabling real-time data access and analysis in various industries.

8 Summary and conclusions

In conclusion, this paper has discussed the importance of secure policies in cloud security frameworks for securing key management services. The introduction highlighted the background, motivation, problem statement, and objectives of the paper. The overview of cloud security provided insights into key concepts and components, while the section on existing frameworks explored popular frameworks used in cloud security. The discussion on secure policies emphasised their significance in key management services, including key lifecycle management, access control, encryption, and key protection. To address the highlighted problem, this research study offers SPCSF, a secure application management framework that makes use of REST API access control. To start, we assess the accuracy of application permissions by comparing the permissions declared with those required, which are obtained from permissions manifests and byte codes. Next, we conduct checks on REST API requests to detect any violations of sensitive API entries, enabling quick identification of the legitimacy of application permissions. Furthermore, we put forward a method for registering and authenticating applications that relies on the K-means clustering algorithm. Finally, we present a dynamic authorization approach for REST APIs that considers the risk levels associated with individual applications. Additionally, case studies demonstrated the application of existing frameworks in a financial institution and a healthcare organization. Finally, potential applications and use cases showcased the versatility of existing frameworks in various industries. By implementing secure policies within cloud security frameworks, organisations can enhance the security and reliability of their key management services, protect sensitive data, and ensure compliance with regulations. Ultimately, the choice to adopt cloud security will hinge on the unique requirements of your business and its security priorities. A plethora of cybersecurity alternatives are accessible, and no single solution can cater to every business. Even within the realm of CC, a variety of options exist, some of which entail integration with your existing conventional solutions. Consequently, it is crucial to contemplate the cybersecurity functionalities imperative for your business, assess the scalability of the solution for your business, and factor in budget constraints before arriving at a determination regarding a cybersecurity solution, whether it is cloud-based or on-premises. In the future, we plan to broaden the application of SPCSF to various controller platforms to showcase its scalability. Additionally, we recognise the significance of investigating the security aspects of controller APIs, and we intend to conduct a comparative study between our proposal and the existing approaches in our future research.

Acknowledgements The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number RI-44-0274.

Author contributions Conceptualization, SA, and SM; data curation, SU and NA; formal analysis, NA, and SA; investigation, SU, and SM; methodology, SA, and NA.; resources, SU and SM; supervision SU; validation, SM and SU; visualization, SM; writing—original draft, SA; writing—review and editing, SM, and SA.

Funding This research work was funded by the Deputyship for Research & Innovation, Ministry of Education, Saudi Arabia through the project number RI-44-0274.

Data availability Not applicable.

Declarations

Competing interests The authors declare that they have no competing interests.

Consent to participate Not applicable.

Consent for publication All authors have read and agreed to the published version of the manuscript.

Ethical approval Not applicable.

References

1. Mell, P., Grance, T.: The NIST definition of cloud computing. Natl. Inst. Stand. Technol. **53**(6), 50 (2011)
2. Sundararajan, E., Arumugam, A., Kumar, S.V.: A study on key management techniques in cloud computing. Procedia Comput. Sci. **115**, 450–457 (2017)
3. Ashwath, A., Kumari, S., Kumar, R.: Security analysis of key management services in cloud computing. Int. J. Comput. Sci. Inf. Security **17**(12), 53–59 (2019)
4. Latif, S., Idrees, Z., Ahmad, J., Zheng, L., Zou, Z.: A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things. J. Ind. Inf. Integr. **21**, 100190 (2021)
5. Awaysheh, F.M., Aladwan, M.N., Alazab, M., Alawadi, S., Cabaleiro, J.C., Pena, T.F.: Security by design for big data frameworks over cloud computing. IEEE Trans. Eng. Manag. **69**(6), 3676–93 (2021)
6. Mayuranathan, M., Murugan, M., Dhanakoti, V.: Best features-based intrusion detection system by RBM model for detecting DDoS in a cloud environment. J. Ambient. Intell. Humaniz. Comput. **12**(3), 3609–3619 (2021)
7. Megouache, L., Zitouni, A., Djoudi, M.: Ensuring user authentication and data integrity in a multi-cloud environment. HCIS **10**(1), 1–20 (2020)
8. Ogonji, M.M., Okeyo, G., Wafula, J.M.: A survey on privacy and security of Internet of Things. Comput. Sci. Rev. **38**, 100312 (2020)
9. Hammami, H., Yahia, S.B., Obaidat, M.S.: A lightweight anonymous authentication scheme for secure cloud computing services. J. Supercomput. **77**, 1693–1713 (2021)

10. El Kafhali, S., El Mir, I., Hanini, M.: Security threats, defense mechanisms, challenges, and future directions in cloud computing. Arch. Comput. Methods Eng. **29**(1), 223–246 (2022)
11. Krishnaveni, S., Vigneshwar, P., Kishore, S., Jothi, B., Sivamohan, S.: Anomaly-based intrusion detection system using support vector machine. In: Artificial Intelligence and Evolutionary Computations in Engineering Systems (pp. 723–731). Springer, Singapore (2020)
12. Mansouri, Y., Babar, M.A.: A review of edge computing: features and resource virtualization. J. Parallel Distrib. Comput. **1**(150), 155–83 (2021)
13. Touqeer, H., Zaman, S., Amin, R., Hussain, M., Al-Turjman, F., Bilal, M.: Smart home security: challenges, issues and solutions at different IoT layers. J. Supercomput. **77**(12), 14053–89 (2021)
14. Chattaraj, D., Bera, B., Das, A.K., Rodrigues, J.J., Park, Y.: Designing fine-grained access control for software-defined networks using private blockchain. IEEE Internet Things J. **9**(2), 1542–1559 (2021)
15. Thakur, S., Chakraborty, A., De, R., Kumar, N., Sarkar, R.: Intrusion detection in cyber-physical systems using a generic and domain-specific deep autoencoder model. Comput. Electr. Eng. **91**, 107044 (2021)
16. Shamshirband, S., Fathi, M., Chronopoulos, A.T., Montieri, A., Palumbo, F., Pescapè, A.: Computational intelligence intrusion detection techniques in mobile cloud computing environments: review, taxonomy, and open research issues. J. Inf. Security Appl. **55**, 102582 (2020)
17. Yoo, S., Jo, J., Kim, B., Seo, J.: Hyperion: a visual analytics tool for an intrusion detection and prevention system. IEEE Access **8**, 133865–133881 (2020)
18. Sarker, I.H.: Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Comput. Sci. **2**(6), 1–20 (2021)
19. Mohammad, A.S., Pradhan, M.R.: Machine learning with big data analytics for cloud security. Comput. Electr. Eng. **96**, 107527 (2021)
20. Lo'ai, A.T., Saldamli, G.: Reconsidering big data security and privacy in cloud and mobile cloud systems. J. King Saud Univ. Comput. Inf. Sci. **33**(7), 810–819 (2021)
21. Narayanan, U., Paul, V., Joseph, S.: A novel system architecture for secure authentication and data sharing in cloud enabled Big Data Environment. J. King Saud Univ. Comput. Inf. Sci. **34**(6), 3121–3135 (2022)
22. Viswanath, G., Krishna, P.V.: Hybrid encryption framework for securing big data storage in a multi-cloud environment. Evol. Intel. **14**(2), 691–698 (2021)
23. Stergiou, C.L., Plageras, A.P., Psannis, K.E. and Gupta, B.B.: Secure machine learning scenario from big data in cloud computing via the internet of things network. In: Handbook of computer networks and cyber security (pp. 525–554). Springer, Cham (2020)
24. Rabbani, M., Wang, Y.L., Khoshkangini, R., Jelodar, H., Zhao, R., Hu, P.: A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing. J. Netw. Comput. Appl. **151**, 102507 (2020)
25. Chiba, Z., Abghour, N., Moussaid, K., Rida, M.: Intelligent approach to build a deep neural network based IDS for cloud environment using combination of machine learning algorithms. Comput. Secur. **86**, 291–317 (2019)
26. Hunt, G., McIlwraith, D.: Cloud Security: A Comprehensive Guide to Secure Cloud Computing. O'Reilly Media, Sebastopol (2019)
27. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput. **13**(18), 1587–1611 (2013)
28. Mather, T., Kumaraswamy, S., Latif, S.: Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Sebastopol (2009)
29. Singh, S., Chaudhary, V.: Cloud computing security issues and challenges: a survey. Int. J. Comput. Appl. **113**(1), 29–36 (2015)
30. Akhtar, N., Vasilakos, A.V.: Security threats in cloud computing. In: Security and Privacy in Communication Networks (pp. 273–299). Springer, New York (2014)
31. Cloud Security Alliance (CSA). (2017). The cloud security guidance. Retrieved from https://cloudsecurityalliance.org/research/security-guidance/
32. Mell, P., Grance, T.: Special publication 800-102, July 2014.
33. Trusted Cloud Initiative (TCI).: Trusted cloud reference architecture. Retrieved from https://trustedcloudinitiative.org/whitepapers/trusted-cloud-reference-architecture/ (2013)
34. Payment Card Industry Security Standards Council (PCI SSC).: Payment card industry (PCI) data security standard (DSS). Retrieved from https://www.pcisecuritystandards.org/document_library (2019)
35. Aljawarneh, S.A., Shuib, L.: Key management techniques in cloud computing: A review study. Journal of Network and Computer Applications, 73, 122–135. [13] Zhao, Z., & Chen, C. (2016). Secure key management and secure access control for cloud storage. Futur. Gener. Comput. Syst. **64**, 155–164 (2016)
36. Ruj, S., Stojmenovic, I., Nayak, A.: Secure data retrieval for decentralized disruption-tolerant military networks using key management. IEEE Trans. Parallel Distrib. Syst. **24**(12), 2429–2443 (2013)
37. Hasan, R., Gouda, M.G., Liu, A.X.: Cryptography in the cloud: A survey. ACM Comput. Surv. **47**(4), 56 (2015)
38. Almorsy, M., Grundy, J., Müller, I.: An analysis of the cloud computing security problem. ACM Comput. Surv. **48**(1), 1–41 (2016)
39. Eiers, W., Sankaran, G., Li, A., O'Mahony, E., Prince, B., Bultan, T. : Quantifying permissiveness of access control policies. In: Proceedings of the 44th International Conference on Software Engineering, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1805–1817. [Online]. Available: https://doi.org/10.1145/3510003.3510233
40. Jayaraman, K., Bjørner, N., Outhred, G., Kaufman, C.: Automated analysis and debugging of network connectivity policies. Microsoft, Tech. Rep. MSR-TR-2014–102, July 2014.
41. Jayaraman, K., Bjørner, N., Padhye, J., Agrawal, A., Bhargava, A., Bissonnette, P.A., Foster, S., Helwer, A., Kasten, M., Lee, I., Namdhari, A.: Validating datacenters at scale. InProceedings of the ACM Special Interest Group on Data Communication 2019 (pp. 200–213), ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 200–213. [Online]. Available: https://doi.org/10.1145/3341302.3342094
42. Fogel, A., Fung, S., Pedrosa, L., Walraed-Sullivan, M., Govindan, R., Mahajan, R., Millstein, T.: A general approach to network configuration analysis. In: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'15. USA: USENIX Association, 2015, pp. 469–483.
43. A. Tang, S. K. R. Kakarla, R. Beckett, E. Zhai, M. Brown, T. Millstein, Y. Tamir, and G. Varghese, "Campion: Debugging router configuration differences," in Proceedings of the 2021 ACM SIGCOMM 2021 Conference, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 748–761. [Online]. Available: https://doi.org/10.1145/3452296.3472925
44. Beckett, R., Gupta, A., Mahajan, R., Walker, D.: A general approach to network configuration verification. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ser. SIGCOMM '17. New York, NY, USA:

Association for Computing Machinery, 2017, p. 155–168. [Online]. Available: https://doi.org/10.1145/3098822.3098834
45. AWS IAM. (2022, Dec.) AWS IAM. [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/access.html
46. Google IAM. (2022, Dec.) Google IAM. [Online]. Available: https://cloud.google.com/iam/
47. AzureRBAC. (2022, Dec.) AzureRBAC. [Online]. Available: https://learn.microsoft.com/en-us/azure/role-based-access-control/overview
48. Kubernetes. (2023, May) Kubernetes RBAC. [Online]. Available: https://kubernetes.io/docs/reference/access-authn-authz/rbac/
49. Casbin. (2022, Dec.) Casbin. [Online]. Available: https://casbin.org/
50. KeyCloak. (2022, Dec.) KeyCloak. [Online]. Available: https://www.keycloak.org/
51. Open Policy Agent. (2022, Dec.) Open Policy Agent. [Online]. Available: https://www.openpolicyagent.org/
52. Padekar, H., Park, Y, Hu, H., Chang, S.: Enabling dynamic access control for controller applications in software-defined networks. In: Proceedings of ACM Symposium on Access Control Models and Technologies, 2016, pp. 51–61.
53. Tripathy, B., Sethy, A., Bera, P., Rahman, M., et al.: A novel secure and efficient policy management framework for software defined network. In: IEEE International Computer Software and Applications Conference (2016), pp. 423–430.
54. Tseng, Y., Pattaranantakul, M., He, R.: Controller DAC: Securing SDN controller with dynamic access control, in: IEEE International Conference on Communications (2017).
55. Xitao Wen, et al.: SDNShield: Reconciliating configurable application permissions for SDN App markets. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN, 2016.
56. Ishakian, V., Muthusamy, V., Slominski, A.: Serving deep learning models in a serverless platform, in: 2018 IEEE International Conference on Cloud Engineering (IC2E), 2018, pp. 257–262. doi:https://doi.org/10.1109/IC2E.2018.00052

Shahnawaz Ahmad is currently an Assistant Professor at Bennet University, Greater Noida, India. He received the B.Tech. degree in Computer Science Engineering and Technology from UPTU, India, in 2012, the M.Tech. degree in Computer Engineering from MDU Rohtak, India in 2013, and a Ph.D. degree in the Faculty of Engineering and Technology Jamia Millia Islamia (A Central University), New Delhi, India in 2023 [NAAC A++ and Ranked 3rd in NIRF 2023]. Having garnered more than 30 research publications in international and national conferences and esteemed journals like Computer Science Review, Cluster Computing, and the Journal of Supercomputing, he has firmly established himself as a prominent figure in the realm of academia. He has acted as the Track Chair for international conferences held in 2023 which were technically cosponsored by IEEE. His specialized knowledge centers around Cloud Computing, Software Engineering, and Machine Learning, with a particular focus on CASB and the development of key management platforms to enhance security within cloud environments.

Shabana Mehfuz (Senior Member, IEEE) is received the B.Tech. degree in Electrical engineering from Jamia Millia Islamia, India, in 1996, the M.Tech. degree in Computer Technology from IIT Delhi, in 2003, and a Ph.D. degree in Computer Engineering from Jamia Millia Islamia, in 2008. She secured First position in the order of merit in B.Tech.(Electrical Engineering). She has been working at the Department of Electrical Engineering, Jamia Millia Islamia, since 1998. She has published more than 130 articles in International/National Journals and Conferences. She has guided eleven Ph.D and is currently supervising ten more candidates. She is a Life Member of ISTE, a member of the Institution of Engineers, and a Life Member of the Computer Society of India. She has received grants for research projects from agencies, such as AICTE and UGC. She has acted as the Track Chair for three flagship international conferences held in 2015, 2019, and 2023 which were technically cosponsored by IEEE. She has been awarded the International Inspirational Women Award 2020 for Best Performer in Government by the GISR Foundation. She has been awarded the National Award for Excellence in Education 2021 in the category of university/college teachers by the AMP organization. Her research interests include Cloud Computing, the Internet of Things (IoT), Wireless Sensor Networks, Mobile ad-hoc Networks, Artificial Intelligence, and Machine Learning.

Shabana Urooj (Senior Member, IEEE) received the B.E. degree in electrical engineering and the M. Tech. degree in electrical engineering (instrumentation and control) from Aligarh Muslim University, Aligarh, Uttar Pradesh, India, in 1998 and 2003, respectively. She obtained her Ph.D. degree in electrical engineering from the Department of Electrical Engineering, Jamia Millia Islamia (A Central University), Delhi, India. She served the industry for three years and teaching organizations for more than 20 years. She is currently working with the Department of Electrical Engineering, College of Engineering, Princess Nourah bint Abdulrahman University, Saudi Arabia. She has authored and co-authored more than 200 research articles, which are published in high-class international journals, reputed conference proceedings and quality books. She has obtained several patents in her name jointly with her colleagues'. She has completed several editorial responsibilities for reputed journals and A class publishers. Dr. Urooj is a recipient of several renowned

international awards; namely Leadership Excellence Women's Award, the Springer's Excellence in Teaching and Research Award, the American Ceramic Society's Young Professional Award, the IEEE's Region 10 Award for Outstanding Contribution in Educational Activities, the Research Excellence Award for quality publishing/authorship and several other national and international academic awards. She is serving as an active volunteer of the Institute of Electrical and Electronics Engineering-IEEE in various capacities and committed to the technological and professional development of society.

Najah Alsubaie received the Ph.D. degree from the Department of Computer Science, University of Warwick, Coventry, U.K. During her Ph.D., she was a member of the Tissue Image Analytics (TIA) Laboratory. She collaborated with several parties, including Coventry Hospitals, where she collected most of the data and received her training in Digital Pathology. She is currently an Assistant Professor at College of Computer & Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. She has published several articles in histology image analysis and has served as a reviewer for several journals.

Authors and Affiliations

Shahnawaz Ahmad[1,2] · Shabana Mehfuz[3] · Shabana Urooj[4] · Najah Alsubaie[5]

✉ Shabana Urooj
smurooj@pnu.edu.sa

Shahnawaz Ahmad
shahnawaz98976@gmail.com

smehfuz@jmi.ac.in

Najah Alsubaie
nmoalsubaie@pnu.edu.sa

[1] Department of Computer Science and Engineering, Mewat Engineering College (WAKF), Haryana, Mewat, India

[2] School of Computer Science Engineering and Technology, Bennett University, Plot Nos 8-11, TechZone II, 201310 Greater Noida, UP, India

[3] Department of Electrical Engineering, Jamia Millia Islamia, Delhi 110025, India

[4] Department of Electrical Engineering, College of Engineering, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia

[5] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, 11671 Riyadh, Saudi Arabia