MLPAM: A Machine Learning and Probabilistic Analysis Based Model for Preserving Security and Privacy in Cloud Environment

Ishu Gupta ©, Member, IEEE, Rishabh Gupta ©, Ashutosh Kumar Singh ©, Senior Member, IEEE, and Rajkumar Buyya ©, Fellow, IEEE

Abstract—The organizational valuable data needs to be shared with multiple parties and stakeholders in a cloud environment for storage, analysis, and data utilization. However, to ensure the security, preserve privacy while sharing the data effectively among various parties have become formidable challenges. In this article, by utilizing encryption, machine learning, and probabilistic approaches, we propose a novel model that supports multiple participants to securely share their data for distinct purposes. The model defines the access policy and communication protocol among the involved multiple untrusted parties to process the owners’ data. The proposed model minimizes the risk associated with the leakage by providing a robust mechanism for prevention coupled with detection. The experimental results demonstrate the efficiency of the proposed model for different classifiers over various datasets. The proposed model ensures high accuracy and precision up to 97% and 100% relatively and secures a significant improvement up to 0.01%, 103%, 151%, 87%, 96%, 43%, and 186% for average probability, average success rate, detection rate, accuracy, precision, recall, and specificity, respectively, compared to the prior works that prove its effectiveness.

Index Terms—Cloud computing, data leakage, data privacy, data security, distribution mechanism, machine learning.

I. INTRODUCTION

DATA storage, analysis, and sharing are the essential services required by any organization to upgrade its performance [1]. Most of the businesses have shifted to the cloud due to its several benefits such as minimum upfront cost and maximum scalability for the required services [2]. However, once the data is transferred for storage and computation purposes in the cloud, the owners lose control over their data [3]. Multiple entities may access the data for commercial and/or other purposes after the data is outsourced [4]. It is not possible to fully trust the cloud platform because it is handled by the third party [5]. Therefore, before uploading data onto the cloud, owners first encrypt their data for privacy reasons. Although some conventional encryption techniques are available for the encryption of owners’ data, such as symmetric and fully homomorphic cryptography, these techniques are insufficient [6], [7]. However, it becomes difficult to perform the computation over the encrypted data [8]. There arises a necessity to protect the owners’ as well as the cloud data while performing the computation effectively. Furthermore, the stored and analyzed data must be shared with the various stakeholders to improve its utility. Although the data is shared among authorized entities, it cannot be assured that data will not be leaked by the receiving entities after obtaining it [9]. Thus, it is essential to protect the data from the entities involved in the communication process. To solve the above-mentioned challenges, we need an effective access control method that supports both the privacy and security of the owners’ data. To the best of the author’s knowledge, no model exists that solves all the aforementioned challenges. In this regard, we propose a novel Machine Learning and Probabilistic Analysis based Model (MLPAM) for data protection through privacy-preserving data storage and analysis, secure sharing, and identification of guilty entity against data leakage in the cloud environment. The main contributions of MLPAM are summarized as follows.

1) To protect the data with enhanced security, all the entities are considered to be untrusted and MLPAM deals with involved entities by effectively defining an access policy.
2) MLPAM enables multiple data owners to freely share the outsourced data. In order to protect the data from stealing or leakage, the data of each owner is encrypted with a separate key and shared in encrypted form.
3) MLPAM uses two clouds where cloud1 deals with data storage, handling, and sharing whereas cloud2 generates the key for the encryption of owners’ data and performs the computation over the data obtained from cloud1 for privacy-preserving classification.
4) An effective distribution mechanism based on an access control is proposed for data distribution among multiple users, that enables to identify the guilty entity and reduces the risk associated with further leakage.
5) A series of experiments are conducted using the widely adopted datasets by researchers to validate the practicality of the proposed model. In addition to this, the comparisons are interpreted among the various a) datasets, b) classifiers, and c) distinctly preprocessed data using \( \epsilon \)-differential...
privacy and with the state of the art works to prove the superiority of MLPAM.

**Organization:** The related work is discussed in Section II. Section III introduces the system and adversary model along with the problem statement and design goals of MLPAM. The proposed model is entailed in Section IV. Sections V and VI describe the applied encryption mechanism and the introduced classification model, respectively. In Section VII, data is distributed based on a distribution factor that is computed using the parameters demanding user sets and data objects sensitivity discussed in Section VII-A. In Section VIII, multiple probabilistic and performance parameters are evaluated. Performance analysis of MLPAM is conducted in Section IX followed by the summary of the proposed work in Section X. Table I depicts the list of notations with their descriptions that have been used throughout the article.

### II. RECENT KEY CONTRIBUTIONS

#### A. Security Based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

Wang et al. [10] proposed a File Hierarchy CP-ABE scheme to secure the data in the cloud environment. This scheme utilized an access structure layered model, which can effectively resist Chosen Plaintext Attacks under the assumption of Decisional Bilinear Diffie–Hellman. The computation cost increased dynamically in this scheme when an integrated ciphertext is computed by the data owner. A data access control scheme for cloud storage to achieve a fair key reconstruction in which none of the users send their shares and no one can access the shared data is proposed by Liu et al. [11]. The experimental analysis demonstrated that computation delay and communication costs are limited, but the authentication is not effective in this scheme. Liu et al. [12] proposed a CP-ABE scheme to reduce the computation cost of the user, as the cost of heavy decryption increases with the complexity of access policy. The performance of the proposed scheme was analyzed by measuring storage overhead and processing power but it lacks in terms of privacy protection. To protect the personal privacy of the user and ensure data confidentiality, Zhang et al. [13] proposed a framework of the Hidden access Policy CP-ABE scheme. They designed an identification method to verify the authorized user and completed the decryption process. This scheme provided a constant size private key independent of the number of user attributes and reduced the transmission as well as storage costs. However, it is considered as a weak security model, because, only the “AND” policy is supported by it. In order to improve the efficiency of the policy and file updation dynamically, Li et al. [14] proposed a CP-ABE scheme based on the linear secret-sharing schemes (LSSS) matrix access structure in cloud computing. This scheme reduced the computing cost of data owner, communication expense, and storage consumption of the proxy cloud service provider as well as resisted the selected plaintext attacks. But, the time cost in file updation is more which is the major downfall of this scheme.

#### B. Privacy-Preserving Machine Learning

A Doubly Permutated Homomorphic Encryption (DPHE) based privacy-preserving mechanism that enabled multiparty protected scalar product is proposed by Yonetani et al. [15], which reduced the high computational cost. The major disadvantage of DPHE is that only one operation either addition or multiplication is supported at a time. Li et al. [16] proposed a scheme for a classifier owner to delegate a remote server and to provide the privacy-preserving classification service for users. A drawback of this scheme is that the interactions of the users were frequently involved while launching a classification query. A data protection scheme is proposed by Li et al. [17], which enabled a trainer to train a Naive Bayes classifier over the dataset provided jointly by different data owners. ϵ-differential privacy is utilized in this scheme to preserve the privacy of every owner. In this approach, the collusion is allowed and adversaries had the ability to forge and manipulate the data. To solve the problem of training the model over the encrypted data under multiple keys, a privacy-preserving deep learning model (PDLM) is proposed by Ma et al. [18]. The model is trained based on stochastic gradient descent and the feed-forward as well as a back-propagation procedure is performed based on a privacy-preserving calculation toolkit. PDLM reduced the storage overhead but the classification accuracy is less and the computation cost is high in this scheme. Li et al. [19] proposed a Privacy-preserving Machine Learning with Multiple data providers scheme to protect the privacy of the datasets. Public key encryption with a double decryption algorithm and ϵ-differential privacy are used to encrypt the datasets of different data providers and the cloud, respectively. However, the proposed solution approached with a high computational cost due to the dependence on integer factorization. Li et al. [20] introduced a privacy-conserving outsourced classification in cloud computing framework under various public keys using fully homomorphic encryption proxy technique. But, the data

<table>
<thead>
<tr>
<th>Table I</th>
<th>List of Terminologies with Their Explanatory Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_O$: data owner; $R_U$: request user; $P_T$: third-party; $C_F$: classifier</td>
<td></td>
</tr>
<tr>
<td>$CP$: cloud platform; $CS$: cloud storage; $CSAP$: cloud service provider</td>
<td></td>
</tr>
<tr>
<td>$CM$: classification model; $D_p$: plain data; $D_{en}$: encrypted data</td>
<td></td>
</tr>
<tr>
<td>$D_{en}$: encrypted data; $D_{pr}$: plain noisy data; $D_{en}$: preprocessed data</td>
<td></td>
</tr>
<tr>
<td>$D_{en}$: training data; $D_{en}$: testing data; $m$: users count; $t$: leaked dataset</td>
<td></td>
</tr>
<tr>
<td>$n$: number of data objects; $k$: third-parties count; $n_c$: count of classes</td>
<td></td>
</tr>
<tr>
<td>$n_{tr}$: training objects count; $n_{te}$: testing objects count; $P_{pub}$: public key</td>
<td></td>
</tr>
<tr>
<td>$P_{priv}$: private key; $SK$: secret key; $N_u$: noise; $N_{en}$: encrypted noise</td>
<td></td>
</tr>
<tr>
<td>$C_{obj}$: object category; $Z_u$: demanding user set; $Y_u$: demanding dataset</td>
<td></td>
</tr>
<tr>
<td>$L_i$: label item; $\lambda$: object categories count; $T_{up}$: tuple; $A_{dp}$: data attribute</td>
<td></td>
</tr>
<tr>
<td>$S_A$: sensitive attribute; $Q_{obj}$: quasi-identifier attribute; $T$: number of tuples</td>
<td></td>
</tr>
<tr>
<td>$A$: attributes count; $A^+_s$: sensitive attributes count; $Z_u$: identity coefficient</td>
<td></td>
</tr>
<tr>
<td>$A^+<em>s$: quasi-identifiers count; $T_j$: attributes of $R_U$, $C</em>{obj}$, object flag</td>
<td></td>
</tr>
<tr>
<td>$TSM_{0}$: formative tuple sensitivity measure; $W_{0}$: cumulative weight</td>
<td></td>
</tr>
<tr>
<td>$TSM_{1}$: cumulative tuple sensitivity measure; $P_{related}$: distribution factor</td>
<td></td>
</tr>
<tr>
<td>$OSM_{0}$: formative object sensitivity measure; $S_{obj}$: sensitivity index</td>
<td></td>
</tr>
<tr>
<td>$OSM_{1}$: cumulative object sensitivity measure; $C_{obj}$: limit coefficient</td>
<td></td>
</tr>
<tr>
<td>$OSM^<em>$: standardized object sensitivity measure; $p^</em>$: worth coefficient</td>
<td></td>
</tr>
<tr>
<td>$P_{select}(X</td>
<td>Y)$: probability of leaking data; $\phi^*$: difference function</td>
</tr>
<tr>
<td>$\overline{\phi}$: average success rate; $\min^{\overline{\phi}}$: detection rate; $N_{COP}$: non-guilty user</td>
<td></td>
</tr>
<tr>
<td>$Q_U$: guilty user; $UE$: untrusted entity; $P_l$: probability; $W_{fp}$: weight factor</td>
<td></td>
</tr>
<tr>
<td>$\theta$: guessing probability; $T_{up}$: threshold value; $CA$: classification accuracy</td>
<td></td>
</tr>
<tr>
<td>$DA$: detection accuracy; $DP$: detection precision; $DR$: detection recall</td>
<td></td>
</tr>
<tr>
<td>$DS$: detection specificity; $T_E$: encryption time; $T_D$: decryption time</td>
<td></td>
</tr>
</tbody>
</table>
owners and the storage servers are considered in the same trustworthy domain that is no longer applicable in the cloud environment. To avoid information leakage under the substitution-then-comparison attack, a scheme was proposed by Gao et al. [21]. By adopting a double-blinding technique to protect data privacy, a privacy-preserving classification mechanism is designed for Naive Bayes and the communication as well as computation overhead are reduced. However, the scheme is not able to achieve the discovery of truth that protects privacy. Hesamifard et al. [22] proposed a framework named CryptoDL for applying deep neural network algorithms to encrypted data. They established neural networking techniques while considering the existing limitations of homomorphic encryption schemes. Although the method works well to secure the private data, the different owners’ data are protected using a key that is not practical.

C. Security Based on Probabilistic Analysis

The pioneering work in the area of probabilistic analysis to detect a guilty agent responsible for leaking the data in a cloud environment named as Guilt Agent Model (GAM) is proposed by Papadimitriou and Garcia-Molina [23]. This model is based on statistical analysis where the probability of various agents for being guilty has been assessed. GAM is widely used by several researchers for malicious user detection in a shared data environment. The parameters for guilty agent detection are improved by Dynamic-Threshold-based Information Leaker Identification Scheme (DT-ILIS) in [24] over GAM [23]. This scheme utilized an access control mechanism to distribute the data among authorized entities. Fan et al. [25] presented a distribution model for data leakage prevention by considering the guilt probability. This model selected a file allocation plan with minimum overlap between obtained file sets of users to find the leakage sources with high probability. In order to share the cloud data in a secure manner, a data leakage detection model (DLDLM) that identified the malicious entity by utilizing an integration of watermarking and probabilistic approach is presented in [26]. To provide stronger security to the shared data, DLDLM utilized the cryptography and hashing techniques and protected the confidential information from the unauthorized entity. The advantage of the probabilistic method is that the leak identification is independent of the alteration or removal in the embedded data, unlike the watermarking technique [23], [27].

The major downfall of the existing work is that the models supported single owners and/or dealt with the single untrusted entity (UE) only, which is not feasible in the real environment. Unlike the existing works, MLPAM establishes a robust mechanism for absolute and efficient data protection in the sharing environment by contemplating all the involved entities as untrusted and ensuring the security and privacy jointly in association with the prevention as well as detection.

III. PROBLEM FORMULATION

This section characterizes the entities involved in the model with their assigned tasks, all the possible threats that may arise in the protocol, defines the problem and outlines the design goals.

A. System Model

The system model comprises the four entities Data Owners (DOid), Cloud Platform (CP), Request Users (RUid), and Third Party (TPid) that are described as follows.

1) **DOid**: An entity generating the information and requesting services from CP. DOid encrypts the data prior to uploading it to CP. Since it is believed that DOid cannot leak its own data, but may leak the other owner’s data, therefore, DOid is treated as an untrusted entity.

2) **CP**: An entity that collects all the encrypted data from DOid and offers storing, computing, and sharing facilities to DOid or RUid. CP transforms the ciphertexts sent by DOid, performs certain computations over it, and encrypts the calculated outcome for secure sharing among DOid or RUid. CP trains the obtained information using machine learning algorithms. CP is a semitrusted or untrusted entity in the model as it follows the protocol strictly, but curious to learn the information. In our system model, CP comprises two clouds where cloud1 consists of Cloud Storage (CS) and Cloud Service Provider (CSP), whereas the Classifier (CF) belongs to cloud2. CSP is the only entity that acts as a bridge and applies ε-differential privacy, distribution mechanism, and detection mechanism to perform the tasks of data transformation, data distribution, and guilty entity detection.

3) **RUid**: An entity receiving the data from CP in the encrypted form along with the key. It obtains the usable data by performing the decryption over the received data from CP. In the system model, RUid is treated as an untrusted entity.

4) **TPid**: An unauthorized and untrusted entity that belongs indirectly to the system. TPid can access the relevant information from a malicious entity or by stealing the dataset from the authorized entity.

B. Adversary Model

CP and RUid are the authorized but untrusted entities in MLPAM having permission to access the data owned by DOid. The following are the possible adversarial threats in MLPAM that can misuse the data through an unauthorized way.

1) **TPid** can corrupt the data owner (DOid; id ∈ [1, n]) for leaking the data of DOid; id′ ∈ [1, n] \ id′ ≠ id.

2) **TPid** can convince Cloud Service Provider (CSP) to leak the data shared by DOid; id ∈ [1, n]. Or TPid can corrupt Classifier (CF) to leak the data shared by CSP.

3) **TPid** can deal with Request User (RUid) in order to leak the data of DOid; id ∈ [1, n] shared by CSP.

4) **TPid** can try to access the data by stealing it during communication among DOid, CSP, CF, and RUid.

5) Third Party **TPid** can acquire the data by stealing it from DOid; id ∈ [1, n].

6) **TPid** can compromise the data by stealing through any malicious activity from cloud1 or cloud2.

7) **TPid** can misuse the data after stealing it from RUid.
C. Problem Statement and Design Goals

The multiple Data Owners DO₁, DO₂, ..., DOₙ possess the data D₁, D₂, ..., Dₙ that need to be shared with CP and among a set of Request Users RU₁, RU₂, ..., RUₘ. The following are the challenges faced by DO₁, DO₂, ..., DOₙ during sharing.

1) Data sharing comprises the risks of security, privacy, and leakage. The entities involved in communication (DO_id, CP, RU_id) can misuse/leak the shared data D₁, D₂, ..., Dₙ or unauthorized third party TP_id can obtain D₁, D₂, ..., Dₙ by stealing from DO_id, CP, and RU_id during communication among DO_id, CP, RU_id.

2) To protect the object Dᵢ from DO_id; id ∈ [1, n] \ id ≠ i.

3) To protect the owners’ data D₁, D₂, ..., Dₙ, it is shared in encrypted form D₁ₑ, D₂ₑ, ..., Dₙₑ, but the computations over D₁ₑ, D₂ₑ, ..., Dₙₑ have limited accuracy.

4) If D₁ₑ, D₂ₑ, ..., Dₙₑ is shared along with the key, then CP or TP_id may obtain D₁, D₂, ..., Dₙ.

The following are the design goals of MLPAM based on the aforementioned problem statement and the adversary model.

1) To provide DO₁, DO₂, ..., DOₙ with an efficient method that allows them to share their data D₁, D₂, ..., Dₙ while preserving security and privacy.

2) To preserve the confidentiality of D₁, D₂, ..., Dₙ via sharing it among the authorized parties only and by protecting the data of DOᵢ from other owners DO_id; id ∈ [1, n] \ id ≠ i, CP, unprivileged RU₁, RU₂, ..., RUₘ, and TP_id. RU₁, RU₂, ..., RUₘ are entitled to access D₁, D₂, ..., Dₙ based on the distribution mechanism.

3) To perform the computation over D₁, D₂, ..., Dₙ while preserving privacy and with improved accuracy and efficiency.

4) To share D₁, D₂, ..., Dₙ by minimizing the likelihood of leakage and to be capable of detecting the guilty user with improved accuracy.

IV. PROPOSED MODEL

The architecture of the proposed model called MLPAM is depicted in Fig. 1. It shows the entities involved along with the communication among them as well as the critical blocks with essential flow among these blocks. Let the owners Ω = {DO₁, DO₂, ..., DOₙ} own the data D = {D₁, D₂, ..., Dₙ} where the data object Dᵢ ∈ D is independent and can be of any type and size. Ω need to share D among authorized parties like Cloud Platform (CP) and various users RU = {RU₁, RU₂, ..., RUₘ} for storage, computation, and performance enhancement, etc., but do not aspire D to be leaked to an unauthorized third party TP = {TP₁, TP₂, ..., TPₖ}. The Classifier (CF) generates private keys PVᵦ = {PV₁K, PV₂K, ..., PVₖK} and public keys PBᵦ = {PB₁K, PB₂K, ..., PBₖK}. The keys PB₁K, PB₂K, ..., PBₖK are shared with Cloud Service Provider (CSP), which transfers these to DO₁, DO₂, ..., DOₙ, respectively. To make the data private and secure, DO₁, DO₂, ..., DOₙ procure the encrypted data DE = {D₁ₑ, D₂ₑ, ..., Dₙₑ} by applying an encryption technique along with their individual key PBₖK, PB₂K, ..., PBₖK. MLPAM utilizes the CP-ABE that distributes the data DE itself as per the without any loss and promises access control via applying decryption to perform the m and produces a Classification Model CM = {CM₁, CM₂, ..., CMₖ} to encrypt and decrypt D₁, D₂, ..., Dₙ, which are obtained via after applying E, Dₙ after leaks the received. Afterward, CSP transforms the stored D₁ₑ, D₂ₑ, ..., Dₙₑ into encrypted noised data Dᴺ = {D¹ᴺ, D²ᴺ, ..., Dₙᴺ} by adding random noise over it and transmits it to CF for computation. The entity CF decrypts D¹ᴺ, D²ᴺ, ..., Dₙᴺ using PV₁⁻¹, PV₂⁻¹, ..., PVₖ⁻¹ individually and obtains the plain noised data Dᴺ = {D¹ᴺ, D²ᴺ, ..., Dₙᴺ} to perform the classification over it. CF performs the computation using D¹ᴺ, D²ᴺ, ..., Dₙᴺ and produces a Classification Model (CM). Any query can be made by DO₁, DO₂, ..., DOₙ or RU₁, RU₂, ..., RUₘ through CSP. The entity CSP communicates with CF, which receives the results from CM and sends back to CSP. Afterward, CSP delivers the acquired results to the corresponding entity DO₁, DO₂, ..., DOₙ or RU₁, RU₂, ..., RUₘ. Furthermore, RU₁, RU₂, ..., RUₘ may request the data from CSP that distributes the data D₁ₑ, D₂ₑ, ..., Dₙₑ among RU₁, RU₂, ..., RUₘ after applying the distribution mechanism. RU₁, RU₂, ..., RUₘ achieve the plain data D₁, D₂, ..., Dₙ via applying decryption over D₁ₑ, D₂ₑ, ..., Dₙₑ along with the corresponding keys PV₁⁻¹, PV₂⁻¹, ..., PVₖ⁻¹, which are obtained via communicating with CF. If any RU_j ∈ RU leaks the received data to any TP₁, TP₂, ..., TPₖ then it is called as Guilty User.

Authorized licensed use limited to: University of Melbourne. Downloaded on August 28, 2021 at 00:54:37 UTC from IEEE Xplore. Restrictions apply.
Algorithms compared are evaluated by statistical analysis, and the $G_U$ is identified by analyzing the evaluated parameters.

V. DATA ENCRYPTION AND DECRYPTION

Let a data owner $DO_i$ has data $D_i$, public keys $PB^K_i$, and defines an access policy $Q_i$ over the attributes. The data object $D_i$ is encrypted with $PB^K_i$ using the following equation:

$$D_i^E = \{Q_i, D_i, D_1, D_2, D_{(x,y)}, D_{(x,y)}^*, D_{(x,y)}^\prime, D_{(x,y),j}^*\}. \quad (1)$$

$D_i$ and $D_1$ are computed by $DO_i$ using the following equation:

$$\hat{D}_i = D_i e(g, g)^{z_{s_i}} \quad D_i^* = g^{z_{s_i}} \quad (2)$$

whereas $D_{(x,y)}$ and $D_{(x,y)}'$ are calculated using the following equation:

$$D_{(x,y)} = h^q(x,y)^{(0)} \quad D_{(x,y)}' = H(\text{att}(x, y))^{q(x,y)^{(0)}} \quad (3)$$

where $Q_i$ is a policy to access the data $D_i$, $e$ is the bilinear map denoted by $e: \mathbb{G}_O \times \mathbb{G}_O \rightarrow \mathbb{G}_A$, $g$ be the generator of $\mathbb{G}_O$, and $\mathbb{G}_O$ be the bilinear group of prime order $p$. For each node $(x, y)$ (including the leaf nodes) in $Q_i$, a polynomial $q(x,y)$ must be chosen from starting with the root node, $s, r, z$ are the random numbers, whereas $a, b$ are the random exponents that belong to $Z_p$ and $h = g^a$. A hash function $H$ is used to map the attributes. $att$ is a function that denotes the attributes within the tree associated with the leaf nodes. $D_{(x,y)}$ for each node $(x, y)$ and $V_j = 1, 2, \ldots, k^*$ is computed to obtain the threshold gate set where $k^*$ is the level of the tree, $DO_i$ do not rely on $CSP$ for data access control and $RU_1, RU_2, \ldots, RU_m$ get different decryption privileges according to their different attributes. The encrypted data $D_i^E$ including the access structure $Q_i$ implicitly is uploaded to $CSP$ by $DO_i, DO_2, \ldots, DO_n$. The attribute sets $Q_j; j = 1, 2, \ldots, m$ are obtained by CSP in the encrypted form along with $SC^K_1, SC^K_2, \ldots, SC^K_m$ from $RU_1, RU_2, \ldots, RU_m$ where $SC^K_1$ denotes the secret key of $RU_j$ related to the attribute set $Q_j, RU_j$ can decrypt the ciphertext $D_i^E$ and gets the original data $D_i$ only if $Q_j$ satisfies $Q_i$. $D_i^E$ is decrypted using the following equation:

$$D_i = \hat{D}_i / \left(e(h^a, g^{(a+r)/b})/e(g, g)^{z_{s_i}}\right). \quad (4)$$

Fig. 2 portrays the encryption and decryption mechanism of MLPAM where $RU_1, RU_2, \ldots, RU_m$ can access $D_1, D_2, \ldots, D_n$ by decrypting the data $D_1^E, D_2^E, \ldots, D_n^E$ using the corresponding keys $PV_1^K, PV_2^K, \ldots, PV_n^K$ after matching of users’ attributes with the access policy determined by $DO_1, DO_2, \ldots, DO_n$.

VI. DATA CLASSIFICATION

To enhance the accuracy and efficiency of the computations while preserving privacy, encrypted data $D_i^E = \{D_1^E, D_2^E, \ldots, D_n^E\}$ from Cloud Storage (CS) is transformed into noised data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ using $\epsilon$-differential privacy [29], [30]. $CSP$ generates a noise vector $N = \{N_1, N_2, \ldots, N_n\}$ using a distribution that is encrypted using public keys $PB^K = \{PB_1^K, PB_2^K, \ldots, PB_n^K\}$ correspondingly and encrypted noise vector $E = \{N_1^E, N_2^E, \ldots, N_n^E\}$ is obtained. The generated data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ are added in the corresponding data $D_i^E = \{D_1^E, D_2^E, \ldots, D_n^E\}$ as $D_i^N = D_i^E + N^E$ where $i \in [1, n]$ and the resulted data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ are passed to $CF$. Using the corresponding private keys $PV_1^K, PV_2^K, \ldots, PV_n^K$, $CF$ decrypts the data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ and attains plain noised data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ that undergoes preprocessing to achieve the preprocessed data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$. Let ith decrypted data $D_i^N$ consists of $\Delta$ attributes $A = \{A_1, A_2, \ldots, A_\Delta\}$, it is preprocessed by using the normalization function given in (5), where $A_1$ is the training sample, $\mu$ and $\sigma$ are the mean and the standard deviation of the training sample, respectively

$$D_i^N = \frac{(A_i - \mu)}{\sigma}. \quad (5)$$

It is known that the data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ belongs to $n^{\star} \leq n$ classes $C = \{C_1, C_2, \ldots, C_n\}$ where $\cup_{i=1}^{n}\cap_{j=1}^{n}C_i = D$ and $\cap_{i=1}^{n}C_i = \emptyset \forall i, j = 1, 2, \ldots, n^{\star} \land i \neq j$. The data $D_i^N = \{D_1^N, D_2^N, \ldots, D_n^N\}$ is divided into training data $D_i^{N_t} = \{D_1^{N_t}, D_2^{N_t}, \ldots, D_{n^{t}}^{N_t}\}$ and testing data $D_i^{N_t} = \{D_1^{N_t}, D_2^{N_t}, \ldots, D_{n^{t}}^{N_t}\}$ satisfying the following properties: 1) $D_i^{N_t} \cup D_i^{N_t} = D_i^N$; 2) $D_i^{N_t} \cap D_i^{N_t} = \emptyset$; 3) $n^{*\prime} \leq n$; and 4) $n^{*\prime} = n \times x, n^{*\prime\prime} = n \times (1 - x)$, where $x \in \mathcal{Z}$ and $0 \leq x \leq 1$ for the Classification Model (CM). The training data $D_i^{N_t}, D_i^{N_t}, \ldots, D_i^{N_t}$ is used to train CM utilizing machine learning algorithms, whereas the testing data $D_i, D_i, \ldots, D_i$ is used to evaluate the accuracy of CM. During the testing process, data objects $D_i^{N_t}, D_i^{N_t}, \ldots, D_i^{N_t}$ are given to CM to identify their classes. CM analyzes $D_i^{N_t}, D_i^{N_t}, \ldots, D_i^{N_t}$, and produces
a Label Vector $L = \{L_1, L_2, \ldots, L_n\}$ as an output, where $L_i \in \mathbb{L}$ specifies $C_i \subset \mathbb{C}$ to which $D_i \in \mathbb{D}$ pertains. The Classification Accuracy (CA) is measured using (6), where $CN$ signifies the number of correctly classified items and $TN$ implies the total number of test items. The stepwise process for classification of data $D_1, D_2, \ldots, D_n$ is depicted in Fig. 3.

$$CA = \frac{CN}{TN}. \quad (6)$$

VII. DATA DISTRIBUTION

Let $\mathbb{D} = \{D_1, D_2, \ldots, D_n\}$ is the dataset consisting $n$ independent data objects in the relational form owned by $n$ different owners $\mathbb{D} \mathbb{O} = \{DO_1, DO_2, \ldots, DO_n\}$, which are stored on cloud storage $CS$ in encrypted form by CSP. The stepwise process along with essential blocks for the data distribution is presented in Fig. 4.

A. Demanding Usersets and Sensitivity Computation

The $m$ distinct users $RU_1, RU_2, \ldots, RU_m$ send the demanding datasets $Y_1, Y_2, \ldots, Y_m$, where $Y_j \subset \mathbb{D}$, $Y_1 \cap Y_2 \neq \emptyset$, $Y_j \cap \mathbb{D} \neq \emptyset$, and $Y_j \cup Y_j \neq \emptyset$. $Y_j$ is used to compute demanding usersets $Z_1, Z_2, \ldots, Z_n$ for each data object $D_i$ using $Z_i = \{RU_j | D_j \in \mathbb{Y}_j\} \forall i = 1, 2, \ldots, n$.

Let the object $D_i$ has $\Omega$ tuples $T = \{T_1, T_2, \ldots, T_\Omega\}$ and $\Delta$ attributes $A = \{A_1, A_2, \ldots, A_\Delta\}$, out of which $\Delta^\ast$ are sensitive attributes $S = \{S_1, S_2, \ldots, S_{\Delta^\ast}\}$. A Grading Function assigns a weight $0 \leq W \leq 1$ to object $D_i$, sensitive attributes $S_\alpha; (\alpha = 1, 2, \ldots, \Delta^\ast)$ of $D_i$, quasi-identifier attributes $Q_\beta; (\beta = 1, 2, \ldots, \Delta^\ast)$ of $D_i$, every possible value $V_\alpha$ of $S_\alpha$, and each possible value $V_\beta$ of $Q_\beta$ as per their sensitivity by satisfying the following properties: 1) $0 \leq W_{D_i} \leq 1$; 2) $W_{S_1} + W_{S_2} + \cdots + W_{S_{\Delta^\ast}} = 0.9$; 3) $W_{Q_1} + W_{Q_2} + \cdots + W_{Q_{\Delta^\ast}} = 0.1$; 4) $0 \leq W_{V_\alpha} \leq 1$; 5) $0 \leq W_{V_\beta} \leq 0.1$, where $W_{V_\alpha}$ and $W_{V_\beta}$ signify the weight of possible values of $S_\alpha$ and $Q_\beta$ attribute, respectively. For a large number of possible values $V_\alpha$ of $S_\alpha$ or $V_\beta$ of $Q_\beta$, $W$ is assigned to $V_\alpha$ or $V_\beta$ by classifying the domain of $S_\alpha$ or $Q_\beta$ relatively. The cumulative weight $W^\ast \in [0, 1] \wedge W^\ast \in \mathbb{R}_{>0}$ for the possible values of $S_\alpha$ and $Q_\beta$ attribute is computed using $W_{V_\alpha} = S_\alpha \times W^\ast$ and $W_{V_\beta} = Q_\beta \times W^\ast$, respectively. The Formative Tuple Sensitivity Measure $TSM^F_{\eta}$ is computed for each tuple $\eta_{\eta} \leq \Omega$ using (7) where $W^\ast_{\eta}$ and $W^\ast_{\beta}$ represent the weight assigned to the $\eta_{th}$ tuple of $S_\alpha$th and $Q_\beta$th attribute, respectively.

$$TSM^F_{\eta} = \sum_{\alpha=1}^{\Delta^\ast} W^\ast_{\eta} + \sum_{\beta=1}^{\Delta^\ast} W^\ast_{\eta} \wedge \forall \eta = 1, 2, \ldots, \Omega. \quad (7)$$

The Cumulative Tuple Sensitivity Measure $TSM^C_\eta \in \mathbb{R}_{>0}$ is obtained by computing the ratio of $TSM^F_{\eta}$ to the Identity Coefficient $I^C_\eta \in [1, \Omega] \wedge I^C_\eta \in \mathbb{Z}_+^*$ as shown in (8), where $I^C_\eta$ is the number of repetitions as a unit of quasi-identifier’s values.
The evaluation of Formative Object Sensitivity Measure $O\!SM^F_{d_i} \in \mathbb{R}_{\geq 0}$ followed by the assessment of Cumulative Object Sensitivity Measure $O\!SM^C_{d_i} \in \mathbb{R}_{\geq 0}$ is performed in (9) and (10), respectively

$$O\!SM^F_{d_i} = \sum_{\eta=1}^{\Omega_i} T\!SM^*_{\eta},$$  

(8)

$$O\!SM^C_{d_i} = W_{d_i} \times O\!SM^F_{d_i},$$  

(9)

The Standardized Object Sensitivity Measure $O\!SM^S_{d_i} \in \mathbb{R}_{\geq 0}$ and $O\!SM^S_{d_i} \in [0, 1]$ is calculated using (11) where $\rho^F \in \mathbb{R}_{\geq 0}$ and $\rho^C \in [0, 1]$ is the Worth Coefficient. $O\!SM^S_{d_i}$ and $\rho^C$ imply the sensitivity of the object $d_i$ and worth of the object allocation, respectively.

$$O\!SM^S_{d_i} = \frac{O\!SM^C_{d_i}}{\max_{i=1,2,\ldots,n} O\!SM^C_{d_i}} + (1 - \rho^C).$$  

(11)

### B. Distribution Mechanism

The objects $d_1, d_2, \ldots, d_n$ are categorized into $\lambda \leq n$ classes $C^* = \{C^1_i, C^2_i, \ldots, C^\lambda_i\}$ having property $\bigcup_{i=1}^{\lambda} C^i_i = D$, $C^i_i \cap C^j_i = \emptyset \forall i, j, i \neq j \in D$ by classifying the range of $O\!SM^C_{d_i}$ completely and disjointly. The range of the $i$th category is $(i-1) \leq \zeta \leq i \zeta$ and $(\lambda-1) \leq \zeta \leq 1$ where $\zeta = \frac{1}{100}$. A Sensitivity Index $S^C_i \in \mathbb{R}_{\geq 0}$ and $S^C_i \in [0, 1]$ using (12) and a Limit Coefficient $\xi^C_i \in \mathbb{R}_{\geq 0}$ and $\xi^C_i \in [0, 100]$ are assigned to the $i$th category $C^i_i \in C^* \forall i = 1, 2, \ldots, \lambda$

$$S^C_i = \left\{ \begin{array}{ll} 1 & \text{if } i = \lambda \times \frac{100}{\lambda - 1} \\ \text{Round} \left[ (i-1) \times \left( 1 - \frac{1}{\lambda - 1} \right) \right] & \text{otherwise} \end{array} \right..$$  

(12)

$C^C_i$ indicates the lower limit in the percentage of users for $d_i$ allocation. $S^C_i$ and $\xi^C_i$ assigned to class $C^i_i$ are $S^C_i$ and $C^C_i$ of all $d_i \in C^i_i$. The Distribution Factor $D^F_i \in [0, |\mathbb{R}|] \cap D^F_i \in \mathbb{Z}_{\geq 0}$ for each $d_i$ is computed employing (13) that defines the count of users for $d_i$ allocation

$$D^F_i = \left[ \min \left( 1, \left( 1 - S^C_i + \xi^C_i \right) \times |Z_i| \right) \right].$$  

(13)

CSP selects the datasets $Y^1, Y^2, \ldots, Y^m$ for distribution among $RU_1, RU_2, \ldots, RU_m$ that minimizes the risks associated with data leakage from $\prod_{i=1}^{m} \left( \frac{1}{|P_i|} \right)$ possible datasets allocations. The operational summary for MLPA distribution data is delineated in Algorithm 1. Object Flag $O^F_i$ is initiated to 1 for every $RU_j$ that indicates the request of $RU_j$ to be processed. The user selection is followed by object allocation and the process is repeated until all the requests are processed. The steps for user selection are depicted in Algorithm 2, where the $n$th user is selected on a rotation basis having $O^F_i$ less or equal to the number of requests by $RU_0$. For the selected user $n$, initiating from object-identity equals object flag to the total number of requests, an object $d_i$ is allocated with a positive data allocation factor, which is reduced by 1 after the object allocation.

---

### Algorithm 1: MLPAM Distribution Mechanism.

**Input:** $n$, $m$, $Y^1, Y^2, \ldots, Y^m$, $Z_i$, $D^F_i$ for $i = 1, 2, \ldots, n$

**Output:** $Y^1, Y^2, \ldots, Y^m$

1: Initialize: $Y^1_j \leftarrow 0$, $Y^2_j \leftarrow 0$, $\ldots$, $Y^m_j \leftarrow 0$.
2: While $\sum_{i=1}^{m} Z_i > 0$
   1. $\mathcal{R} \leftarrow$ SELECT_USER($m, |Y|, O^F$).
   2. For $i \notin \emptyset$: $Y^i_j \leftarrow Y^i_j + 1$.
   3. If $D^F_i \leq \{Y^i_j\}$ then $O^F_j \leftarrow 1$ for $j = 1, 2, \ldots, n$
   4. Break
3: End if
4: End do
5: Return $\mathcal{R}$

---

### Algorithm 2 MLPAM User Selection.

**Input:** $n$, $m$, $Y^1, Y^2, \ldots, Y^m$

**Output:** $\mathcal{R}$

1: Initialize: $\mathcal{R} = 1$, $\mathcal{R}^* = 0$
2: Function SELECT_USER($m, |Y|, O^F$)
3: If $\mathcal{R} = \mathcal{R} + 1$ then $\mathcal{R} = 1$
4: If $O^F_i \leq \{Y^i_j\}$ then $\mathcal{R}^* = \mathcal{R}^* + 1$
5: Return $\mathcal{R}^*$
6: End function

---

### VIII. GUILTY USER DETECTION

If any $RU_j$ leases the dataset $\ell \in D$ to an unauthorized party $TP_k \in TP$, then the following parameters are calculated for the identification of Guilty User $GU(j)$: 1) Probability ($P_b$) of $RU_j$ for being $GU$ ($P_b(GU|\ell)$) for $j = 1, 2, \ldots, m$; 2) difference function ($\varphi(j,k)(GU)$) for $j = 1, 2, \ldots, m$; 3) average success rate ($\omega^*_{(j,k)}$); 4) detection rate ($\omega^*$) using (14)-(17), respectively [24], where $\theta$ is the probability of stealing any $d_i \in \ell$ by $TP_k \in TP$ either from $DO_i \in D^C$ or $RU_j \in D^F$ having access to $D$. It is believed that the obtained $d_i \in D$ by $TP_k$ in $TP$ in case of 1) leakage through $DO_i \in D^C$ where $i \neq j$, $CSP$, and $CF$ 2) stealing from cloud1, cloud2, and communication among $DO_i \in D^C$. CSP $\in CP$, $CF$ $\in CP$, and $RU_j \in D$ will not be usable since it is in encrypted $D^F$ (using distinct public keys $PB_R^1, PB_R^2, \ldots, PB_R^n$) or noised $D^N$ form

$$P_b(GU|\ell) = 1 - \prod_{d_i \in e_\ell} \left( 1 - \left( 1 - \theta \right) \frac{D^F_i}{D^F_{\ell}} \right)$$  

(14)

$$\varphi^*_{(j,k)}(GU) = P_b(GU|\ell) - P_b(GU|\ell)$$  

(15)

$$\omega^* = \sum_{j,k=1,2,\ldots, m} \omega^*_{(j,k)}(GU)$$  

(16)

$$\min \omega^* = \min_{j,k=1,2,\ldots, m} \omega^*_{(j,k)}(GU).$$  

(17)

A $RU_j \in \mathbb{R}$ fulfilling the criteria $\max_{j=1,2,\ldots, m} P_b(GU|\ell) - P_b(GU|\ell) \leq T_H$ is declared as $GU$ where $T_H$ signifies the threshold value. Furthermore, the parameters Detection
Accuracy (DA), Detection Precision (DP), Detection Recall (DR), and Detection Specificity (DS) are computed using (18)–(21), respectively, to prove the effectiveness of MLPAM by classifying the users as Guilty User ($G_U$) and Nonguilty User ($NG_U$) and via assessing the outcome of MLPAM, whether the user is guilty or nonguilty. In these equations, $|NG_U^A = NG_E^A|$ specifies the number of test cases having Actual Nonguilty Users ($NG_U^A$) equal to the Estimated Nonguilty Users ($NG_E^A$). $|G_U^A = G_E^A|$ is the term representing the count of test cases where Actual Guilty Users ($G_U^A$) are identical to the Estimated Guilty Users ($G_E^A$). $|NG_U^A = G_E^A|$ demonstrates the number of test cases in which actual guilty users are estimated as nonguilty users, whereas $|NG_U^A = G_E^A|$ deals with the number of test cases when actual nonguilty users are estimated as guilty users.

$$DA = \frac{|NG_U^A = NG_E^A| + |G_U^A = G_E^A|}{|NG_U^A = NG_E^A| + |G_U^A = G_E^A| + |NG_U^A = G_E^A| + |NG_U^A = G_E^A|}$$

(18)

$$DP = \frac{|G_U^A = G_E^A|}{|G_U^A = G_E^A| + |NG_U^A = G_E^A|}$$

(19)

$$DR = \frac{|G_U^A = G_E^A|}{|G_U^A = G_E^A| + |G_U^A = NG_E^A|}$$

(20)

$$DS = \frac{|NG_U^A = NG_E^A| + |NG_U^A = G_E^A|}{|NG_U^A = NG_E^A| + |NG_U^A = G_E^A|}$$

(21)

The computational and space complexities are $O(\max_{1 \leq j \leq m}[Y_j]), O(1); O((n^{*})^3), O((n^{*})^2); O(\sum_{j=1}^{m}[Y_j]), O(\sum_{j=1}^{m}[Y_j]); O(m^2 \max_{1 \leq j \leq m}[Y_j]), O(m^2)$ for various phases data encryption and decryption, data classification, data distribution, and guilty user detection of MLPAM, respectively. The complexity analysis of MLPAM implies that the data is protected by the aid of endurable time and space, which establishes its potency.

IX. PERFORMANCE EVALUATION

A. Experimental Setup

A series of experiments have been conducted over four different datasets Glass, Iris, Wine, and Balance Scale with 10, 4, 13, 4 attributes and 214, 150, 178, 625 instances, respectively, that are taken from the UCI Machine Learning Repository [31] to train CM using machine learning algorithms. The four different classifiers Support Vector Machine (SVM), Random Forest, K-Nearest Neighbor (K-NN), and Naive Bayes have been used to train CM over the training data. These experiments are performed on Intel Core i7-7700 CPU@3.60 GHz eight-core processor with Ubuntu 14.04-amd64 operating system, 8 GB RAM machine using Python 2.73 for encryption and machine learning, and C++ 12.1.2 for guilty user detection.

B. Computation Time for Encryption/Decryption

The computation time for the encryption and decryption processes over the various datasets is shown in Fig. 5(a) and (b), respectively. It is observed that the time costs of the encryption and decryption grow linearly with respect to the number of attributes associated with the access policy. Furthermore, the comparison among different datasets has been performed in terms of encryption time ($T_E$) and decryption time ($T_D$). It is found that both encryption and decryption time varies with respect to the dataset for the fixed number of attributes. For instance, BS has maximum encryption time for 2 and 5 attributes, but it is not true for other numbers of attributes. However, Iris has minimum encryption time for 2 and 3 attributes, which is not true for 4 and 5 attributes. BS has minimum decryption time for all number (2–5) of attributes, whereas wine has maximum decryption time for all attributes excluding 3. CP-ABE is effective to reduce the computation cost because the user can determine the matching result without the interaction with the initiator.

C. Accuracy of Classification Model

From the complete dataset, 9/10 of the data is used as training data, whereas the rest of the data is taken as testing data. The machine learning is performed over both clean and noised data. To generate the noised data, we have used the Gaussian and the randomly generated mechanisms with the value of privacy level 0.1. The outcome of the noised data is compared against the clean data to find the variations. Furthermore, a comparison is performed among Gaussian and Random noised data to find the superior one. The outcome of $CM$ is measured using the testing data and the Classification Accuracy ($CA$) is computed. Fig. 6(a)–(d) shows the $CA$ achieved by $CM$ of MLPAM over Clean, Gaussian noised, and Random noised data and also depicts the comparison among Glass, Iris, Wine, and BS datasets for SVM, Random Forest, KNN, and Naive Bayes classifier, respectively. It is observed that $CA$ of the noised data is less compared to the clean data in the case of all the four classifiers because of the noise addition but still, $CA$ is nearly equal for noised data and also provides more security compared to the clean data. Furthermore, out of the two noised added data, the Gaussian noised data outperforms over the Random noised data in the case of all the four classifiers. The performance of datasets and classifiers in descending order are Iris, Wine, BS, Glass; and SVM, Random Forest, Naive Bayes, K-NN, respectively. Out of the four datasets, the Iris dataset outperforms the rest of the three dataset for all the four classifiers. For the Clean and Gaussian noised data, the SVM classifier outperforms the
rest three, whereas for the Random noised data, Naive Bayes outperforms over the other classifiers. As an aggregate, the SVM classifier outperforms the rest three classifiers in MLPAM due to the application of kernel trick and considerable optimal margin gap between separating hyperplanes during classification, which results in better performance.

D. Parameters for Guilty User Detection

Five hundred data objects are shared among ten users and the requests of \( RU_1, RU_2, \ldots, RU_m \) are generated randomly in MLPAM. \( \lambda = 11, C^*_i = 0.1 \forall C^*_i \in C^*, \theta \in \{0.1, 0.3, 0.5\}, \) and \( T^*_H = 0.0E + 00, 0.0E + 00, 1.0E + 00, 7.75E - 05 \) has taken into consideration throughout the experiments. The performance is assessed with respect to the **Weight Factor**, which is calculated as \( W_F = \sum_{j=1}^{m} Y_j \). The experimental results are compared with GAM [23] and DT-ILIS [24] via implementing these on the same platform.

The average probability \( \frac{\sum_{j=1}^{m} P_k(G_U | Y_j)}{|RU|} \) when all \( RU_j \) have leaked their allocated datasets \( Y_1, Y_2, \ldots, Y_m \). Average Success Rate (\( \bar{\tau}^* \)), and Detection Rate (\( \bar{\tau}^* + \bar{\omega}^* \)) for the proposed and the comparable schemes are computed with respect to \( W_F \) at different \( \theta = 0, 0.1, 0.3, 0.5 \) in Tables II–IV, respectively. \( \sum_{j=1}^{m} P_k(G_U | Y_j) \) is noted for all three GAM [23], DT-ILIS [24], and MLPAM \( \forall W_F \) and \( \theta = 0, 0.1, 0.3, 0.5 \), whereas the values of \( \sum_{j=1}^{m} P_k(G_U | Y_j) \) are depicted in Table II for \( \theta = 0.3, 0.5 \). The following are the observations from Table II.

1. Probability to detect \( G_U \) is very high \( \forall W_F, \theta \).
2. \( \frac{\sum_{j=1}^{m} P_k(G_U | Y_j)}{|RU|} \) decreases with respect to \( \theta \) as chances of stealing rather than leaking the data become high with increment in \( \theta \).
3. Probability of the proposed and compared schemes is nearly the same, but, in the proposed scheme, the difference between the probabilities of \( G_U \) and \( NG_U \) is high (see Tables III and IV) that makes the scheme capable to identify \( G_U \) with high accuracy.

In Tables III and IV, the values of \( \bar{\tau}^* \) and \( \bar{\omega}^* \) decrease with respect to \( W_F \) since the overlapping among the datasets \( Y_1, Y_2, \ldots, Y_m \) raise with increment in \( W_F \). Furthermore, \( \bar{\tau}^* \) and \( \bar{\omega}^* \) increase with respect to \( \theta \) due to increment in the probabilities difference of \( G_U \) and \( NG_U \). The Detection Accuracy (\( DA \)), Detection Precision (\( DP \)), Detection Recall (\( DR \)), and Detection Specificity (\( DS \)) achieved by MLPAM with respect to \( \theta \), and comparison against [23], [24] are shown in Fig. 7(a)–(d), respectively. MLPAM secures \( DA 80\%, 97\%, 97\%, 96\%, DP 72\%, 95\%, 100\%, 99\%, DR 100\%, 99\%, 93\%, 92\%, \) and \( DS 60\%, 95\%, 100\%, 99\% \) for \( \theta = 0, 0.1, 0.3, 0.5 \), respectively, that are very high and acceptable over existing methods [23], [24]. An average for all the three parameters \( \frac{\sum_{j=1}^{m} P_k(G_U | Y_j)}{|RU|} \), \( \bar{\tau}^* \), and \( \bar{\omega}^* \) is calculated individually for each \( \theta \). The improvement attained by MLPAM over [23] and [24] for each parameter individually with respect to \( \theta \) is depicted in Table V. MLPAM achieves relative improvement up to 0.0088064%, 102.83%, 151.31%, 86.54%, 96.08%, 43.08%, 185.71% for \( \frac{\sum_{j=1}^{m} P_k(G_U | Y_j)}{|RU|}, \bar{\tau}^*, \bar{\omega}^* \), \( DA, DP, DR \), and \( DS \), respectively, which supports its effectiveness. Moreover, Table VI depicts the comparison of complexities and it is indicated that both the computational and space complexities are the least in MLPAM as compared to GAM [23] and DT-ILIS [24] due to the effectual data allocation strategy in the proposed model. Additionally, we have performed a comprehensive feature analysis along with a comparison of MLPAM against the state of the artworks [1], [13], [18], [23], [24]. It can be seen from

![Fig. 6. Accuracy of CM in MLPAM for (a) SVM, (b) Random Forest, (c) K-NN, and (d) Naive Bayes classifier.](image-url)
Table VII that MLPAM is the only model that synchronously supports multiple untrusted entities, owners, users, and ensures as well as significantly enhances several indispensable features simultaneously; therefore, MLPAM performance is better than the existing models [1], [13], [18], [23], [24].

### Security Analysis

In our system model, all the authorized entities DO_{id}, RU_{id}, CSP, CF, and an unauthorized entity TP_{id} are deemed as untrusted; and MLPAM protected the data from every involved entity. To protect the data of an owner DO_{id}; \ id \in [1, n] against a) leakage by other owners DO_{id}'; \ id' \in [1, n] \land \ id' \neq \ id or CSP to TP_{id} b) stealing by TP_{id} from cloud1/DO_{id}; \ id \in [1, n] or during communication among DO_{id}, CSP, CF.

## TABLE III

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>( W_{s} )</th>
<th>( P_{s} )</th>
<th>( \rho )</th>
<th>( \rho^{*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>GAM [23]</td>
<td>4.63812E+01</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
</tr>
<tr>
<td>0.5</td>
<td>GAM [23]</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
<td>6.3516E-01</td>
</tr>
</tbody>
</table>

## TABLE IV

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>( W_{s} )</th>
<th>( P_{s} )</th>
<th>( \rho )</th>
<th>( \rho^{*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>GAM [23]</td>
<td>4.63812E+01</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
</tr>
<tr>
<td>0.5</td>
<td>GAM [23]</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
<td>6.3516E-01</td>
</tr>
</tbody>
</table>

### TABLE V

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>( W_{s} )</th>
<th>( P_{s} )</th>
<th>( \rho )</th>
<th>( \rho^{*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>GAM [23]</td>
<td>4.63812E+01</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
</tr>
<tr>
<td>0.5</td>
<td>GAM [23]</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
<td>6.3516E-01</td>
</tr>
</tbody>
</table>

### TABLE VI

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>( W_{s} )</th>
<th>( P_{s} )</th>
<th>( \rho )</th>
<th>( \rho^{*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>GAM [23]</td>
<td>4.63812E+01</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
</tr>
<tr>
<td>0.5</td>
<td>GAM [23]</td>
<td>6.9136E+01</td>
<td>6.3516E-01</td>
<td>6.3516E-01</td>
</tr>
</tbody>
</table>
and $RU_{id}$, the data is encrypted with a distinct key and shared in encrypted form. Furthermore, for protecting the data against leakage by $CF$ or stealing by $TP_{id}$ from cloud2, while performing the analysis with high accuracy, the data is shared in noised form and MLPAM achieved a significant classification accuracy up to 92%. However, if any $RU_{id}$ leaks the data intentionally to $TP_{id}$ or somehow, $TP_{id}$ becomes successful in stealing the data from $DO_{id}; id \in [1, n]/RU_{id}$, then the data is protected through leaker identification via performing probabilistic analysis. The experimental results signified that MLPAM is capable of recognizing a $G_U$: effectively by securing up to 1 $P_k$, 0.791441 $\pi^*$, 0.390156 min $\pi^*$, 97% $DA$, 100% $DP$, 100% $DR$, and 100% $DS$ that validates its robust security.

X. CONCLUSION AND FUTURE WORK

This article proposed a novel model named MLPAM for effective data protection in a real Cloud environment. To provide the stronger security, all the involved entities are considered to be untrusted and a robust mechanism is provided in the model by exploring every possible threat that may arise during data flow among the involved parties. MLPAM presented an effective sharing protocol to mitigate the loss due to data leakage. An influential distribution mechanism is proposed for data allocation and to detect a guilty entity with high confidence. The evident experimental results depicted that the guilty entity can be distinguished easily in the proposed scheme, which proves its effectiveness. MLPAM attained a significant improvement up to 186% over the existing works and simultaneously secured significant Detection Accuracy, Precision, Recall, and Specificity compared to the prior works that support its high performance. The comprehensive analysis and performance of the model over the well-known datasets and comparison with the existing works demonstrated that MLPAM is more secure, efficient, and optimal. MLPAM lays a foundation for future secure and efficient data sharing and management in multiple environments like Internet of Things, Big Data, etc. Furthermore, the request users might become capable of acquiring the data objects that are not allocated among these users through the use of shared keys. The emerged issue is referred to as future work and can be resolved by employing the set of distinct keys for the data objects.

REFERENCES


Rishabh Gupta received the MCA degree in computer science from the Guru Jambheshwar University of Science and Technology, Hisar, India, in 2015. He is currently working toward the Ph.D. degree in computer science from National Institute of Technology, Kurukshetra, Kurukshetra, India. His research interests include cloud computing, deep learning, big data, and parallel processing.

Ashutosh Kumar Singh (Senior Member, IEEE) received the Ph.D. degree in electronics engineering from the Indian Institute of Technology BHU, Varanasi, India, in 2000. He is currently a Professor and Head with the Department of Computer Applications, National Institute of Technology, Kurukshetra, Kurukshetra, India. He has more than 20 years of research and teaching experience in various universities in India, U.K., and Malaysia. He is a Postdoctoral Researcher from the Department of Computer Science, University of Bristol, Bristol, U.K. He has authored/coauthored more than 250 research papers and 8 books. His research interests include verification, design and testing of digital circuits, data science, cloud computing, machine learning, security, etc.

Rajkumar Buyya (Fellow, IEEE) received the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002. He is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne, VIC, Australia. He is also the Founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in cloud computing. He has authored more than 650 publications and seven textbooks including Mastering Cloud Computing published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese, and international markets, respectively. He is one of the highly cited authors in computer science and software engineering worldwide (H-index=138, G-index=307, over 100000 citations).