Minimax Theory and its Applications
Volume 11 (2026), No. 3s, 714 to 733

MFedL-MTD: Enhancing Cloud Security via Multi-level Federated Learning-based Optimal Task Offloading

R.Udaya Nirmala Mary[1], Dr.R.Santhosh[2]

[1]Research scholar, Department of Computer Science and Engineering Faculty of Engineering Karpagam Academy of Higher Education Coimbatore, Tamil Nadu, India, Email: udaya.7it@gmail.com
[2]Professor and Head Department of Computer Science and Engineering Faculty of Engineering Karpagam Academy of Higher Education Coimbatore, Tamil Nadu, India, Email: santhoshrd@gmail.com

Abstract
In the realm of cloud security, optimal task offloading entails the strategic assignment of computational tasks to cloud resources, bolstering efficiency, scalability, and resilience in the face of potential security vulnerabilities.During task offloading in cloud computing, security concerns encompass data privacy, the potential for sensitive information leaks, and the risks posed by unauthorized access to cloud resources. Henceforth, optimal task offloading is essential to enhance cloud security as it optimizes resource utilization, reduces latency, and minimizes attack surfaces, thus mitigating potential security risks. Recent works are majorly focused on task offloading however, researches are limited user privacy and handling of cloud server overflowing issues. To overcome these issues, we introduced MFedL-MTD framework which encompasses Multi-level Federated Learning with Moving Target Defense (MTD) technique for enhancing task offloading security. In this work, cloud user's authenticity is ensured by performing Multi-factor Authentication (MFA) thereby authenticating legitimate users via Trust Authority (TA). Following that, Multi-level Federated Learning is adapted for executing effecting user selection in edge server. Here, local model, global model and super model are generated and aggregated by utilizing Federated Learning Secure Aggregation (FL-SA) which improves accumulation performance. Moreover, task classification is implemented using N-beats and then task scheduling is accomplished through considering appropriate features. At last, MTD is performed for VM placement and VM migration when cloud server is overflows thus tends to effective and secure task offloading. The proposed MFedL-MTD framework can be evaluated using various metrics as latency, computation complexity, security strengthen, migration time and energy consumption alongside comparisons with state-of-the-art approaches, demonstrating its superior performance over existing models.

Keywords: Cloud Security, Task Offloading, Multi-level Federated Learning, Multi-factor Authentication, Federated Learning Secure Aggregation.

Introduction

Cloud security involves safeguarding data, applications, and infrastructure hosted on cloud platforms. It employs measures like encryption, access controls, and multi-factor authentication to protect against unauthorized access and data breaches [1]. Cloud providers implement robust security protocols, including firewalls and monitoring systems, ensuring the integrity and confidentiality of stored information. Regular audits, compliance checks, and automated security updates contribute to a proactive defense against evolving cyber threats [2]. Organizations leveraging cloud services must also implement security best practices and educate users to ensure a collaborative effort in maintaining a resilient and secure cloud environment. Securing the cloud is crucial as it safeguards sensitive data and applications from cyber threats. With businesses increasingly relying on cloud services, ensuring robust security prevents unauthorized access, data breaches, and potential disruptions [3]. Cloud security measures protect against evolving cyber threats, compliance violations, and data loss. It instills trust among users and clients, enhancing overall data integrity and confidentiality. A secure cloud environment is essential for maintaining business continuity, fostering innovation, and upholding the reputation and trust of organizations relying on cloud infrastructure.User and entity authentication play a pivotal role in ensuring the security of cloud computation, where sensitive data and applications are hosted remotely [4-6]. Authentication verifies the identity of users and entities interacting with the cloud, preventing unauthorized access and potential security breaches. Firstly, robust user authentication is essential to confirm the legitimacy of individuals accessing cloud resources. This involves multi-factor authentication (MFA) techniques such as passwords, biometrics, or security tokens, adding layers of protection against unauthorized access

attempts. Additionally, entity authentication ensures that devices and services communicating with the cloud are legitimate and not compromised, reinforcing overall security and privacy [7].

Moreover, Mobile Threat Defense (MTD) based offloading enhances cloud security by detecting and mitigating potential threats originating from mobile devices [8]. MTD solutions analyze device behaviour and network activities to identify anomalies and potential security risks. Offloading certain processing tasks to the cloud, after ensuring the legitimacy of the user and entity, enables real-time threat detection and response, enhancing the overall security posture [9]. However, securely implementing MTD-based offloading presents challenges and limitations. Privacy concerns arise as offloading involves transmitting data to the cloud for analysis. Ensuring compliance with data protection regulations becomes crucial to address these privacy issues [10-12]. Furthermore, the latency introduced by offloading may impact the performance of real-time applications, necessitating careful consideration of the trade-offs between security and system responsiveness. Additionally, the potential for false positives or negatives in threat detection poses a challenge [13]. MTD solutions may incorrectly flag legitimate activities as threats or fail to identify sophisticated attacks. Continuous updates and improvements in threat detection algorithms are essential to mitigate these limitations and enhance the accuracy of MTD-based offloading [14]. In conclusion, user and entity authentication, coupled with MTD-based offloading, are integral components in securing cloud computation. While they significantly bolster the overall security posture, addressing privacy concerns and fine-tuning threat detection mechanisms remain crucial to achieving a robust and balanced approach to cloud security [15]. Regular updates and a proactive security stance are essential for adapting to evolving cyber threats and maintaining the integrity of cloud-based systems.

Authentication is a crucial aspect of security in any system, including cloud computing. However, it also comes with its limitations. One limitation is the risk of unauthorized access due to weak authentication mechanisms [16]. If authentication protocols are not robust enough, attackers may exploit vulnerabilities to gain access to sensitive data or resources. This underscores the importance of employing strong authentication methods such as multi-factor authentication (MFA) to mitigate this risk.Another limitation is the overhead associated with authentication processes [17]. Authentication adds computational overhead and latency to system operations, especially in large-scale cloud environments where numerous users and devices need to be authenticated concurrently. This overhead can impact system performance and scalability, potentially leading to delays in service delivery or increased resource consumption.Furthermore, authentication mechanisms may suffer from vulnerabilities such as brute-force attacks, where attackers attempt to guess authentication credentials through repeated login attempts [18]. To address this limitation, it is essential to implement measures like account lockout policies or rate limiting to prevent such attacks.Optimal task scheduling in cloud computing involves efficiently allocating computational resources to tasks to minimize execution time and resource utilization while meeting performance requirements. However, this process is challenging due to several factors.

One limitation is the dynamic and unpredictable nature of cloud environments. Workloads may vary over time, and resource availability can fluctuate due to factors like demand spikes or hardware failures. Consequently, traditional task scheduling algorithms may struggle to adapt to these changes effectively [19].Additionally, the complexity of modern cloud applications and their interdependencies pose challenges for task scheduling. Tasks may have dependencies on each other, requiring careful coordination to ensure correct execution order while maximizing resource utilization [20].Furthermore, achieving optimal task scheduling often involves solving NP-hard optimization problems, which may require significant computational resources and time to find the best solution, especially in large-scale cloud environments.To address these limitations, researchers are developing advanced task scheduling algorithms that leverage techniques such as machine learning, genetic algorithms, and heuristics to adaptively schedule tasks based on real-time conditions and application requirements. Additionally, leveraging techniques like containerization and serverless computing can provide more flexibility and efficiency in task execution and resource allocation in cloud environments.

Research Contribution

To perform optimal task offloading, we have contributed several innovations in our proposed MFedL-MTD framework which are detailed as follows,

The MFedL-MTD framework integrates Multi-level Federated Learning with Moving Target Defense (MTD) techniques to enhance task offloading security, offering a robust solution for protecting sensitive data during cloud-based computation.

By employing Multi-factor Authentication (MFA) and a Trust Authority (TA), cloud user authenticity is ensured, mitigating the risk of unauthorized access and bolstering the overall security posture of the system.

Leveraging Federated Learning Secure Aggregation (FL-SA) for model generation and aggregation, along with task classification using N-beats and MTD for VM placement and migration, the framework optimizes task offloading effectiveness while maintaining robust security measures, particularly in scenarios of cloud server overflow.

Literature Survey

In this section, we reviewed state-of-art in enhancing of cloud security which is further divided into three sub-sections which are detailed as follows:

Authentication Approaches for cloud user

Author in [21] proposes a user authentication protocol leveraging Elliptic Curve Cryptography (ECC) for ensuring security in cloud computing environments. ECC offers strong security with relatively smaller key sizes compared to other cryptographic schemes, making it suitable for resource-constrained cloud environments. The protocol likely addresses issues such as authentication efficiency, resistance against common attacks, and scalability. This research [22] presents an anonymous authentication scheme tailored for cloud users, emphasizing both computational efficiency and security. Anonymous authentication schemes are crucial for protecting user privacy while accessing cloud services. The paper [23] may focus on techniques such as cryptographic primitives or zero-knowledge proofs to achieve anonymity while ensuring efficient authentication. Author in [24] introduces a user authentication scheme for cloud environments that incorporates three factors to enhance security while preserving user anonymity. Multi-factor authentication (MFA) is increasingly essential for thwarting various cyber threats. This scheme likely combines factors like passwords, biometrics, tokens, or other authentication mechanisms to strengthen security.This study [25] presents a novel authentication scheme specifically designed for mobile cloud computing environments, blending graphical authentication methods with traditional authentication mechanisms. Mobile devices often have unique characteristics and constraints, necessitating specialized authentication approaches. The hybrid scheme might combine graphical patterns, PINs, or biometrics to authenticate users securely on mobile platforms.

This paper [26] proposes a comprehensive framework for two-factor authentication (2FA) in cloud computing, emphasizing security against various threats. 2FA is widely adopted to add an extra layer of security beyond traditional password-based authentication. The framework could encompass factors such as possession (e.g., tokens or smartphones) and knowledge (e.g., passwords or PINs), with considerations for usability and resilience against attacks.Author in [27] explores the concept of attribute-based pseudonymity to achieve privacy-preserving authentication in cloud services. Pseudonymity allows users to interact anonymously while still being able to access services securely. Attribute-based mechanisms enable access control based on user attributes, enhancing privacy and security in cloud environments. The paper [28] may delve into techniques like attribute-based encryption or anonymous credentials.This paper introduces a user authentication protocol leveraging blockchain technology to preserve privacy in distributed mobile cloud environments. Blockchain offers decentralized and tamper-resistant storage, suitable for securing sensitive authentication data [29]. The protocol likely employs techniques such as cryptographic hashing, digital signatures, or smart contracts to ensure privacy and integrity in authentication processes.Each of these research papers contributes to the field of user authentication in cloud computing environments, addressing various aspects such as security, efficiency, anonymity, and usability. By reviewing and analyzing these works collectively, researchers can gain insights into the current trends, challenges, and potential solutions in this critical area of cloud security.

Secure Offloading in Cloud Environments

This paper [30] introduces a user authentication protocol leveraging blockchain technology to preserve privacy in distributed mobile cloud environments. Blockchain offers decentralized and tamper-resistant storage, suitable for securing sensitive authentication data. The protocol likely employs techniques such as cryptographic hashing, digital signatures, or smart contracts to ensure privacy and integrity in authentication processes.This research [31] focuses on resource management strategies for secure computation offloading in softwarized cyber-physical systems. With the increasing complexity and interconnectedness of cyber-physical systems, efficient resource allocation is crucial for ensuring security and performance. Author in [33] explore techniques such as dynamic resource provisioning, workload scheduling, and security-aware allocation to optimize offloading processes.This study proposes a secure computation offloading scheme for blockchain-based IoT networks, incorporating deep reinforcement learning (DRL) techniques. IoT networks often face resource constraints and security challenges, making efficient offloading strategies essential. By leveraging DRL, the scheme may adaptively learn optimal offloading decisions while considering security constraints and resource availability.This paper [34] presents a secure computation offloading approach inspired by blockchain principles

for vehicular cloud networks. Vehicular environments demand efficient offloading strategies to support diverse applications while ensuring data privacy and integrity. The proposed scheme may leverage distributed consensus mechanisms, cryptographic techniques, and decentralized storage to secure computation offloading tasks in vehicular networks.This research addresses secure task offloading in blockchain-enabled mobile edge computing environments using deep reinforcement learning techniques. Mobile edge computing integrates edge devices with cloud resources to support computation-intensive tasks closer to end-users.

Author in [35] may propose a framework that combines blockchain-based security mechanisms with DRL-driven offloading decisions to optimize performance and security.This study introduces a secure and energy-efficient computational offloading mechanism using Long Short-Term Memory (LSTM) networks in mobile edge computing. Energy efficiency is critical in mobile edge environments to prolong device battery life and reduce operational costs. The proposed approach likely employs LSTM networks to model task characteristics and predict optimal offloading decisions while considering security requirements.This paper [36] investigates profit maximization strategies for security-aware task offloading in edge-cloud environments. In commercial settings, maximizing profits while ensuring security is a primary concern for service providers. The research may propose algorithms or frameworks that balance profit maximization with security considerations, potentially incorporating techniques such as game theory or optimization methods.This study explores advanced deep learning techniques for resource allocation and security-aware data offloading in industrial mobile edge computing scenarios. Industrial environments often involve sensitive data and stringent security requirements. The paper [37] may introduce deep learning models tailored for optimizing resource allocation and offloading decisions while meeting security constraints in industrial edge computing settings.By reviewing and analysing these research papers collectively, researchers can gain insights into the state-of-the-art techniques and challenges in secure computation offloading, resource management, and security-awareness in distributed environments such as mobile edge computing, IoT networks, and vehicular networks.

Virtual Machine Placement

This paper [38] proposes a framework for efficiently placing virtual machines in a cloud data center while considering security alongside other objectives. The authors likely analyze various factors such as resource utilization, energy efficiency, and security metrics to devise an optimized placement strategy.This research focuses on securing the allocation of virtual machines in multitenant cloud databases. The SA2-MCD architecture likely addresses concerns related to isolation between tenants, data privacy, and access control within the cloud database environment.This paper [39] introduces a security-aware virtual machine placement approach leveraging hesitant fuzzy decision-making processes. Hesitant fuzzy sets allow for handling uncertainty in security requirements and constraints during the placement of virtual machines, ensuring a robust and secure cloud environment.Blockchainbus presents a lightweight framework for ensuring the security of virtual machine migration in cloud federations. By leveraging blockchain technology, this framework likely addresses issues such as trust, integrity, and provenance during the migration process, enhancing overall security in cloud environments.This paper [40] proposes a secure storage mechanism for cloud environments based on data dispersion and encryption techniques. By dispersing data across multiple storage nodes and encrypting it, the mechanism likely aims to mitigate risks associated with data breaches and unauthorized access in cloud storage systems.CryptDICE introduces a distributed data protection system designed to ensure secure cloud data storage and computation. The system likely employs cryptographic techniques to protect data confidentiality, integrity, and availability while facilitating secure computation in cloud environments.Overall, these research papers collectively contribute to advancing the state-of-the-art in secure virtual machine placement and cloud security mechanisms, addressing various challenges such as data privacy, access control, and integrity assurance in cloud computing environments.

MFedL-MTD Framework

In this study, we proposed MFedL framework for performing secure task scheduling using MTD. Figure 1 represents the overall architecture of proposed MFedL-MTD framework. The framework's application within various organizations is outlined through the following sub-sections description of its techniques:

Multi-factor Authentication

The Multi-Factor Authentication (MFA) is segmented into two stages.The registration procedure and the authentication process. Below are the procedures encompassed by each step:

Cloud User Registration Phase

Throughout the registration process, you will be required to furnish the TA with confidential credentials such as your password (CU_{pass}), biometrics (CU_{bio}), and CU identification (CU_{id}). The ensuing steps outline the process:

Step 1: At the outset, the CBM received a registration request (reg^{req}) from the CU_J. Upon acceptance, the exchanges personal credentials such as CU_{id}, CU_{pass}, and $CU_{bio}$with the TA. Subsequently, he replies to the CU_J with a set of (CU_{id}, CU_{pass}, CU_{bio}).

Step 2: Upon receiving the stacks, the TA initially verifies whether the user is new or already registered by examining the CU_ID and BIOM_ID. If the user is identified as new, then TA computed as,

$$TA = M\left(CU_{id}^{CU_J} \left\| CU_{bio}^{CU_J} \right\| RAM \| Pub^{CU_J}\right), \qquad (1)$$

Here, TA selects a random number, and provides it to the CU_J. The CU_J extracts the de-hashed information and then hashes it, allowing it to be represented as,

$$CU_J = M\left(CU_{id}^{CU_J} \left\| CU_{bio}^{CU_J} \right\| RAM \| Pub^{CU_J}\right). \qquad (2)$$

Cloud User Authentication Phase

TA verifies the registration credentials to ensure the legitimacy of CU_J before communicating with the cloud services. It's important to note that the entire login process takes place on a public channel to facilitate the detection of attackers. The steps involved in the authentication phase are as follows:

Step 1: Initially, the registered CU_J inputs their CU_{id}, CU_{pass}, CU_{bio} onto the cloud server, which is overseen by the CBM. By validating the password and ID, the CBM discerns whether the registered user is authentic.

$$\left(Reg\left[CU_{id}, CU_{pass}, CU_{bio}\right] \overline{=}_{\neq} OR^{*}\left[CU_{id}, CU_{pass}, CU_{bio}\right]\right) \quad (3)$$

If theCU_{id}, CU_{pass} and CU_{bio} match, they may proceed; otherwise, they will be denied access to the network.

Step 2: The authenticated CU_J is now represented as $CU_J = \{APL_{cert^t} \| RAM^2 \| CU_{bio} \| Se_k\}$, where Se_k denotes the session key for secure communication, RAM^2 signifies the second random number, CU_{bio} represents the CU's biometric identity, and APL_{cert^t} stands for the approval certificate for the t-th timestamp. It's crucial to note that each transaction enables the verification of the session key.

Step 3: Subsequently, TA can verify the CU_J biometric data by assessing the uniqueness of the CU_{bio} alongside the freshness of the approval certificate (APL) using the formula,

$$CBM = [(APL_{cert^t} \| RAM^2 \| CU_{bio}) \quad = (APL_{cert^t} \| RAM^2 \| CU_{bio})] (4)$$

If both the CU_J authenticity and freshness align, the interaction concludes, and the CU is deemed valid. Otherwise, with regret, the CU may be expelled from the network. The proposed solution effectively managed the malicious network traffic within the cloud environment, leveraging the authentication mechanism outlined earlier.

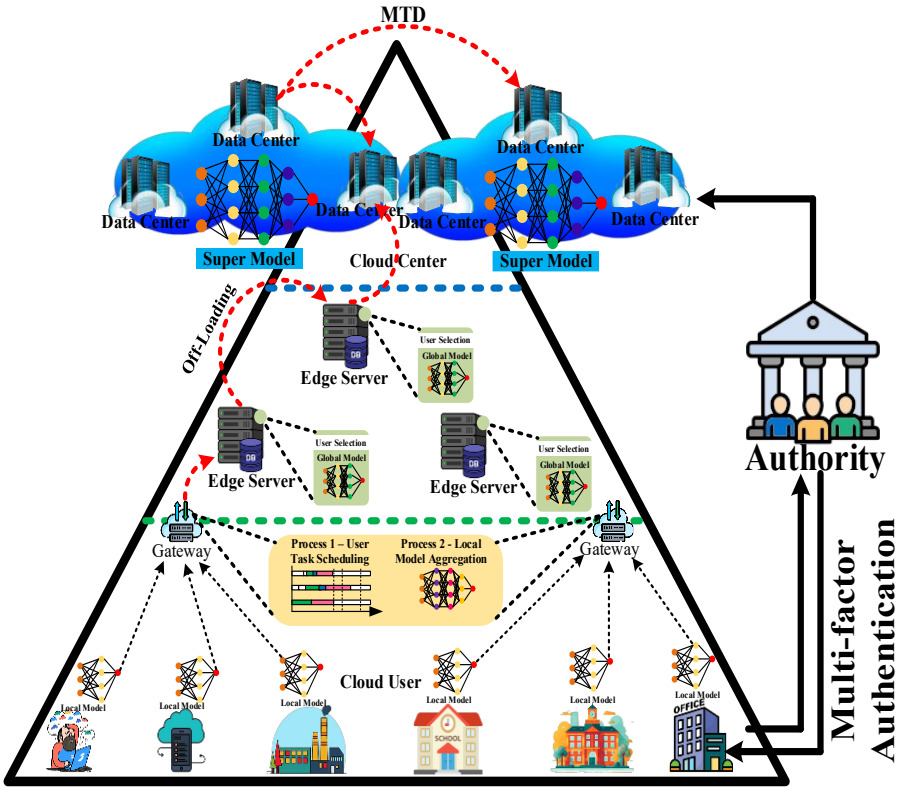


Fig 1. Overall Architecture of Proposed MFedL-MTD

Multi-level Federated Learning based User Selection

Cloud user ecosystem exclusively grants access to authorized their devices. To elaborate, sensing capabilities are restricted to end user devices associated with authorized cloud users. To bolster user privacy, we've integrated homomorphic solutions and multi-level federated learning technique with Federated Learning with Secure Aggregation (FL-SA) as shown in fig 2. Here, the users request their specific task which is generated into local model. Here, based on the local model edge sever selects the optimal task as device availability, computational capability, network connectivity, and data relevance to ensure effective participation and contribution to the collaborative model training process. At last, the global models are aggregated into super global model. The task seeking between the cloud server and users devices consists of multiple rounds. In each round, cloud users devices receive a global model from the cloud server. Subsequently, they train and encrypt their local models, compute decryption dividends, and conduct secure aggregation. This iterative process continues until the desired convergence is achieved. The following provides a comprehensive breakdown of the proposed multi-level federated learning-based data exchange protocol.

The public key parameters of each cloud users are first initialized by the server based on its authenticity, $\mathfrak{D}(\mathfrak{Ч}, \mathscr{E}, \mathrm{Y}, \kappa, \Omega)$, where Ω is the random vector $\Omega \leftarrow \mathrm{U}(\mathrm{R}_{\mathfrak{Ч}}^{\mathbb{D}})$, Y and κ over R represent the distribution of error, $\mathcal{E}$ is the key distribution parameter, and $\mathfrak{Ч}$ is the modulus of the cipher text. Every cloud users$\mathbb{D}_j$ models a secret key $\mathrm{secr}_{kj} \leftarrow \mathcal{E}$ based on the public key parameters; from there, the public key may be calculated using the formula $\mathfrak{PK}_j = -\mathrm{secr}_{kj}.\Omega + \mathfrak{e}_j(\mathrm{mod}\mathfrak{Ч})$. With that, the collaborative public key is computed by all of the authorized cloud users as follows:

$$\widetilde{\mathfrak{PK}} = \sum_{j=1}^{n} \mathfrak{PK}_j = \sum_{j=1}^{n} -\mathrm{secr}_{kj}.\Omega + \sum_{j=1}^{n} \mathfrak{e}_j(\mathrm{mod}\mathfrak{Ч}) \qquad (5)$$

After computing the collaborative public key, every Cloud usersdevice$\mathbb{D}_j$ retrieves the current weights of the global model W_t from the cloud server for each round $\mathfrak{r}$. Using the FL-SA technique, each device constructs its local model with these weights. This addition addresses dimensionality concerns and enhances channel weights, thereby improving overall performance.

Local Model Generation: Generation of the Local Model begins with the FL-SA receiving task request by the cloud usersdevices. Federated Learning with Secure Aggregation (FL-SA) is a technique FL-SA combines federated learning with secure aggregation techniques to enable collaborative model training across decentralized devices while preserving data privacy through encrypted model updates.In FL-SA, secure aggregation is used to aggregate model updates from multiple devices in such a way that individual updates remain encrypted and private.The mathematical equation for FL-SA can be represented as follows:

Let N denote the total number of decentralized users participating in the federated learning process and $\in_i^t$represent the model parameters (weights) of the i-th device at iteration t. Furthermore, $\nabla\in_i^t$represent the global model parameters at iteration t.Assume,$\nabla\in_i^t$ represent the update of the model parameters (weights) for the i-th device at iteration t. Here, $\mathrm{En}(.)$denote an encryption function and $\mathrm{De}(.)$denote a decryption function.In FL-SA, the process of securely aggregating the model updates from the devices can be mathematically represented as:

Step 1: Each device icomputes its model update $\nabla\in_i^t$ based on its local data and the current global model θ^t.Device i encrypts its model update using a public key, $\mathrm{En}(\nabla\in_i^t)$.

Step 2: The encrypted model updates from all devices are aggregated by a central server or aggregator.Moreover, the aggregator securely computes the aggregated update $\nabla\overline{\in}^t$ using secure aggregation techniques, ensuring that individual updates remain encrypted during the aggregation process.

Step 3: Then aggregator sends the aggregated update $\nabla\overline{\in}^t$ to all users.Each user i decrypts the aggregated update using its private key:

$$\nabla\in_i^t = \mathrm{De}(\nabla\in_i^t) \qquad (6)$$

Step 4: Each device iupdates its local model parameters:

$$\in_i^{t+1} = \in_i^t + \nabla\overline{\in}^t \qquad (7)$$

Step 5: At last, updated local model parameters $\in_i^{t+1}$are then used for the next round of training.

This process repeats iteratively until convergence is achieved or a predefined stopping criterion is met. The specific encryption, decryption, and secure aggregation techniques used in FL-SA can vary depending on the implementation and security requirements.Equations (9) to (11) provided above clarify that $\beth_p$represents thepth feature map. $\mathfrak{W}_2$ and $\mathfrak{W}_1$denote the dimensionality weights, $\mathfrak{W}ei$ signifies the channel weight, and δ_{ch}stands for the channel statistics. The primary capsule layer takes the feature maps generated by the squeeze excitation layer and utilizes them to construct the primary capsule, which holds information in vector format. Subsequently, the digit capsule layer accepts these vector-format capsules as input to produce the final output. Within the digit capsule, a squashing function is integrated to facilitate vector-based routing. Let's define the route between thejth and ith capsules as:

$$\hat{\beth}_{i|j} = w_{ji}\beth_j \qquad (8)$$

$$\mathfrak{W}ei_i = \sum_j C_{ji}\,\hat{\beth}_{i|j} \qquad (9)$$

$$\mathbb{o}_i = \frac{\|\mathfrak{W}ei_i\|^2}{1+\|\mathfrak{W}ei_i\|^2}\frac{\mathfrak{W}ei_i}{\|\mathfrak{W}ei_i\|} \qquad (10)$$

Equations (8) to (10) delineate the matrix weight w_{ji} elucidating the relationship between the jth and ith capsules. The connection coefficient is represented byC_{ji}, while the output squashing function $\mathbb{o}_i$provides the perceived outcomes for each individual patient. Following several epochs of local model derivation, the Cloud users furnish the weights of each distinct local model as$wei_h^j, j \in \{1,2,\dots,n\}$.

Encrypting the Local Model Weights-A straightforward representation of the plaintext input for wei_h^j can be denoted as $\mathfrak{F}_j \in H$. Now, let's explore$\Omega = \Omega[0], \widetilde{\mathfrak{P}\mathfrak{K}} = \widetilde{\mathfrak{P}\mathfrak{K}}[0]$.The samples acquired from the public key parameters are $\theta^{d_i} \leftarrow \mathcal{E}\circ e_\eta^{d_i}, e_t^{d_i} \leftarrow \gamma$. For the local model, the ciphertext can be computed as follows:

$$\mathrm{CipT}^{d_j} = \left(\theta^{d_i}, \widetilde{\mathfrak{P}\mathfrak{K}} + \mathfrak{F}_j + e_0^{d_i}, \theta^{d_i}.\Omega + e_1^{d_i}\right)((\mathrm{mod}\daleth)) \quad (11)$$

As per the equation provided, every Cloud usersdevice establishes communication with the cloud server through the encrypted local model, expressed as$\mathrm{CipT}^{\mathrm{d_j}} = \left(\mathrm{CipT}_0^{\mathrm{d_i}}, \mathrm{CipT}_1^{\mathrm{d_i}}\right)$.

Perform Homomorphic Operation: Upon reception of all encrypted local model weights from the Cloud users, the cloud server initiates the homomorphic encryption process by summing up these weights. One expression of this homomorphic operation on local model weights is as follows:

$$CipT_{sum} = \sum_{j=1}^{\eta} CipT^{d_i} = \left(\sum_{j=1}^{\eta} CipT_0^{d_i}, \sum_{j=1}^{\eta} CipT_1^{d_i}\right) \triangleq_{(CipT_{sum_0}, CipT_{sum_1})} (mod Ð) \quad (12)$$

Here, $CipT_{sum}$ represents the aggregate result of the homomorphic operations on the cipher texts.

Decryption Dividends Computation- Each Cloud usersdevice receives the homomorphically encrypted total of the cipher texts, which it utilizes to compute the decryption dividends and subsequently transmits them back to the cloud server. The decryption dividends can be expressed using the following formulation:

$$d_i = sec_{ki}.CipT_{sum_1} + \epsilon_j^* = sec_{kj}.\sum_{j=1}^{\mathcal{N}} \left(\theta_i.\Omega + \epsilon_1^{d_i}\right) + \epsilon_1^* (mod Ð) \quad (13)$$

Secure Global Aggregation- After acquiring each decrypted dividend from the Cloud users, the cloud server proceeds with secure aggregation to obtain the plaintext. One approach to formulate secure aggregation is as follows:

$$\sum_{i=1}^{\mathcal{N}} \mathfrak{F}_j \approx CipT_{sum_0} + \sum_{i=1}^{\mathcal{N}} d_i (mod Ð) \quad (14)$$

The cloud server decrypts$\sum_{i=1}^{\mathcal{N}} \mathfrak{F}_j$ using the aforementioned equation to calculate the average local weights

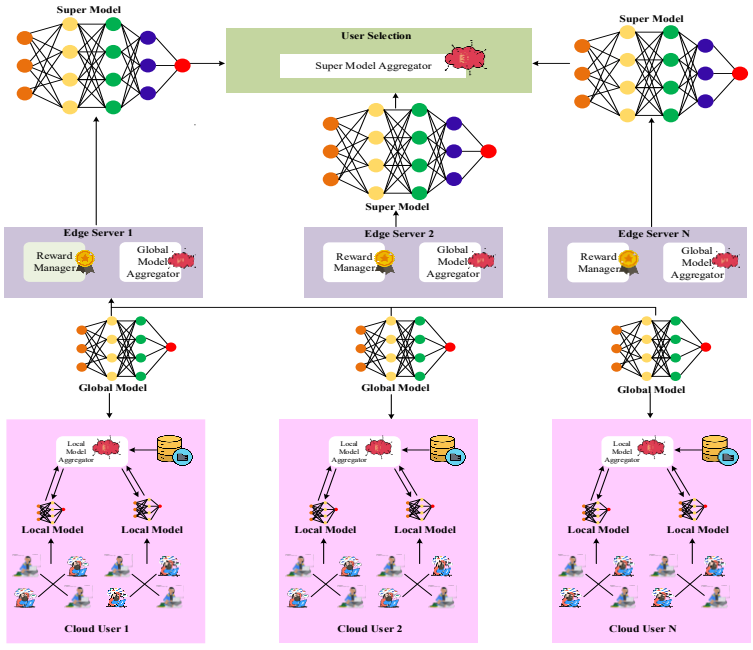


Fig 2. Multi-level Federated Learning

and obtain new global model weights, wei_{t+1}, for subsequent rounds.

$$wei_{t+1} = \frac{1}{\mathcal{N}} \sum_{j=1}^{\mathcal{N}} wei_t^j \quad (15)$$

The communication rounds will continue to increase until the desired convergence is attained. Likewise, the super model is generated and aggretaed.

N-beats based Task Scheduling& Offloading

Let $\mathfrak{y} = \{\gamma\ 1, \gamma\ 2, \gamma\ 3, \cdot\ \cdot\ \cdot, \gamma_{\mathfrak{T}}\}$ denote the number of task request raised by cloud users. Here, $\gamma_t = (\gamma_t^1, \gamma_t^2, \gamma_t^3, \gamma_t^4)$ represents the four-dimensional task request in the cloud environment employed in this study, comprising task type, task latency, cost, reliability, make span, scalability, throughput, reliability and resource utilization, the task classification is performed using N-beats. The prediction task can be summarized as follows: For any given time index t, $\mathfrak{y}_t = \{\gamma_{t+1}, \gamma_{t+2}, \gamma_{t+3}, \ldots\ldots \gamma_{t+G}\}(t = 0,1,2, \ldots.)$,where G denotes the size of the time window, is utilized to predict the future vector $x_{t+n} = \gamma_{t+G+n}$, where n is a non-negative integer. We are primarily concerned with identifying anomalies and making short-term predictions given the pressing nature of the sewage treatment issue. To predict a single data unit following the current time frame, we therefore assign n = 1. Thus, one can view N-BEATS as a form of multiple regression that undergoes repetitive iterations and incorporates nonlinearity. The simplicity in implementation directly stems from the simplicity in concept.

In fig. 3, the complete deep architecture of N-BEATS is constructed using identical Basic Block modules, depicted in blue. The input data γ_t for the entire model serves as the input for Block 1, representing the initial Basic Block. The outputs are two vectors, $\hat{x}_t$ and $\hat{\gamma}_t$. Subsequent blocks receive the residuals of the preceding block as inputs. The Basic Block yields both a backcast of the current point within the forecast window and a forecast of the future point in the forecast window.The detailed internal structure of the Basic Block module is depicted on the far left. It comprises two separate linear layers and a shared fully connected (FYC) network, which generate two forward coefficients $\vartheta_{t,p}^{h,\mathfrak{f}}$ and two backward coefficients $\vartheta_{t,p}^{h,\mathfrak{b}}$ respectively, constituting the Basic Block. The straightforward approach is illustrated in Equations (1) through (6).

$$\mathfrak{G}_{t,p,1}^{h} = FYC_{t,1}(\mathfrak{y}_t), \quad (16)$$

$$\mathfrak{G}_{t,p,2}^{h} = FYC_{t,2}\left(\mathfrak{G}_{t,p,1}^{h}\right), \quad (17)$$

$$\mathfrak{G}_{t,p,3}^{h} = FYC_{t,3}\left(\mathfrak{G}_{t,p,2}^{h}\right), \quad (18)$$

$$\mathfrak{G}_{t,p,4}^{h} = FYC_{t,4}\left(\mathfrak{G}_{t,p,3}^{h}\right), \quad (19)$$

$$\vartheta_{t,p}^{h,\mathfrak{f}} = Lir_{p}^{\mathfrak{f}}\left(\mathfrak{G}_{t,p,4}^{h}\right), \quad (20)$$

$$\vartheta_{t,p}^{h,\mathfrak{b}} = Lir_{p}^{\mathfrak{b}}\left(\mathfrak{G}_{t,p,4}^{h}\right), \quad (21)$$

Taking the calculation of $\mathfrak{G}_{t,p,1}^{h}$and $\vartheta_{t,p}^{h,\mathfrak{f}}$as illustrations, Equations (9) and (10) offer detailed instructions for computing the fully connected layer and the linear layer. Here, $\mathfrak{W}ei$ denotes the weight and Bi represents the bias term.

$$\mathfrak{G}_{t,p,1}^{h} = FYC_{t,1}(\mathfrak{y}_t) = RELU\left(\mathfrak{W}ei_{t,1}\mathfrak{y}_t + Bi_{t,1}\right), \quad (22)$$

$$\vartheta_{t,p}^{h,\mathfrak{f}} = Lir_{p}^{\mathfrak{f}}\left(\mathfrak{G}_{t,p,4}^{h}\right) = \mathfrak{W}ei_{t,p}^{h,\mathfrak{f}}\mathfrak{G}_{t,p,4}^{h}. \quad (23)$$

After applying Equations (9) and (10), the forward and backward coefficients undergo processing through a base layer, resulting in forward and backward outputs denoted as $\hat{x}_{t,p}^{h} = k_{t,p}^{h,\mathfrak{f}}\left(\vartheta_{t,p}^{h,\mathfrak{f}}\right)$ and $\hat{\gamma}_{t,p}^{h} = k_{t,p}^{h,\mathfrak{b}}\left(\vartheta_{t,p}^{h,\mathfrak{b}}\right)$.

$$\hat{x}_{t,p}^{h} = \sum_{\mathrm{i}=1}^{di\left(\vartheta_{t,p}^{h,\mathfrak{f}}\right)} \vartheta_{t,p,i}^{h,\mathfrak{f}}\, \mathfrak{v}_{p,i}^{h,\mathfrak{f}}, \quad (24)$$

$$\hat{\gamma}_{t,p}^{h} = \sum_{i=1}^{di\left(\vartheta_{t,p}^{h,\mathfrak{b}}\right)} \vartheta_{t,p,i}^{h,\mathfrak{b}}\mathfrak{v}_{p,i}^{h,\mathfrak{b}}, \quad (25)$$

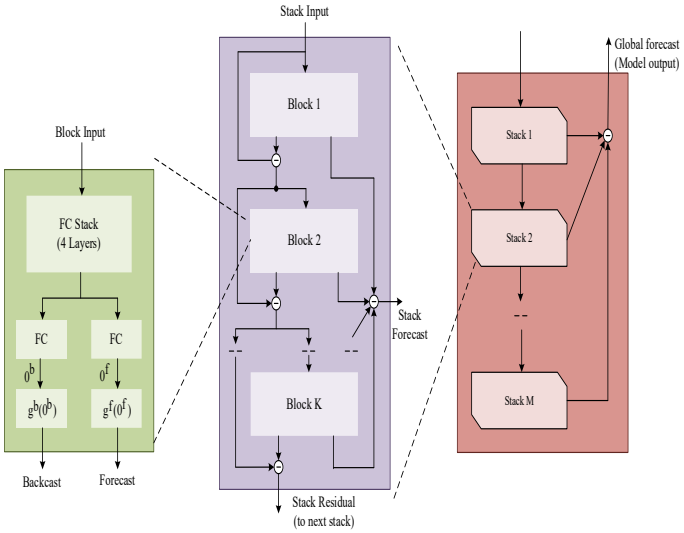


Fig 3. Illustration of N-beats

In this context,$\vartheta^*_{t,p,i}$ represents the coefficient of the basis, and $\mathfrak{v}^*_{p,i}$ signifies the basis itself. The traditional residual connection entails summing the model's input and output and then passing the result to the following block. Following the utilization of Equations (11) and (12), residual connections are established for both the forecasting and backcasting branches.

$$\gamma^h_{t,p} = \gamma^h_{t-1,p} - \hat{\gamma}^h_{t-1,p}, \qquad (26)$$

$$\hat{x}_{t,p} = \sum_{h=1}^{H} \hat{x}^h_{t,p}, \qquad (27)$$

$$\hat{x}_{t+1} = \sum_{p=1}^{P} \hat{x}_{t,p}, \qquad (28)$$

Subsequent blocks may prioritize addressing the segments that remain ambiguous due to the residuals from the backcast section, hindering their assimilation of the preceding block's knowledge. Within the stack, the residuals of the forecasted portion are initially amalgamated, followed by aggregation at the overarching level of the network (stack aggregation). The anticipated value, represented by $\hat{x}_{t+1}$in Equation (13), serves as the ultimate output of the network. Based on task scheduling, the proposed MFedL-MTD offloading task can be done in three levels: (1) The user can perform tasks if the task and device constraints are matched (ii) If the task constraints are overloaded than the device capacity, the task can be offloaded to edge server, and (iii) if the edger server gets overloaded, the task can be offloaded to cloud server. Furthermore, in our proposed work, we performed VM placement and VM migration in secure manner via MTD technique while the cloud server is overflowed.

Experimental Results

In this section, we demonstrate the detailed experimental setup with implementation results based on validation metrics and comparisons with existing works are offered.

Implementation Setup

The implementation of the proposed work is carried out using MATLAB simulation tool in the HP laptop with processor of AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz frequency rate, 8GB Random Access Memory (RAM), 1TB SSD memory with windows 11 Operation System (OS). Since we perform secure offloading in the cloud environment in three levels (i.e. cloud-cloud, cloud-edge, and cloud-gateway), the simulation parameters and multi-level federated learning parametersof the proposed environment are provided in table I below,

Table 1. Simulation Parameter

Simulation Metrics	**Description**
# of IoT Devices	60 to 100
# of Edge Server	15 to 70
# of Cloud Server	5
# of Gateway	3
Incoming Task Rate	40 task/sec
Bandwidth of the Channel	25 MHZ
Frequency of CPU in Cloud Devices	120/113 cyc/sec
Frequency of CPU on IoT Devices	400×108 cyc/sec
Frequency of CPU in Edge Server	60×115 cyc/sec
Usage of Energy	0.45 Joules
IoT Devices Tx Power	mW
Type of Network	Randomized
Federated Learning Parameters	
Simulation Parameters	**Description**
Learning Rate	0.003, 0.3, 0.003
Aggregation of Super Model	2
Aggregation of Global Model	4
Epochs for Local Training	3,6,6
Processing Deadline	5,35,35 (sec)
Workload Computation	3.42, 29.4,12.7 (Mbytes)
Update Size of Local Model	0.29, 19.8, 34.4 (Mbits)

Furthermore, we have also provided the details of simulation parameters of the adopted N-Beats algorithm in the table II below,

Table 2. N-Beats Simulation Parameter

Hyperparameters	**Value**
# of Epochs	25
Batches/Epoch	55
Loss	P-MAPE
Learning Rate	0.002
Optimizer	Adam
Size of Batch	512
Period of Lookback	15
No. of Layers & Blocks	5 & 5
Width	512

Performance and Comparative Results

This sub-section compares the performance of the proposed secure offloading model with the existing works such as DRL-BSCO [9], MEC-STO [11], LSTM-Off [12], and PM-STO [13] with the validation metrics such as latency, computational complexity, security strength, migration time, and energy consumption respectively. The detailed explanation for the metrics is provided below,

Latency Analysis

In offloading scenario, the latency is defined as the transferring delay of computational tasks from a local cloud device to the remote server for task execution. Mathematically, the latency in offloading task can be formulated by summating the various time delay factors as,

$$\mathrm{Lat}_{\mathrm{off}} = \mathrm{DP}^{\mathrm{Lat}} + \mathrm{TO}^{\mathrm{La}} + \mathrm{Se}^{\mathrm{La}} + \mathrm{DTx}^{\mathrm{La}} + \mathrm{Ne}^{\mathrm{La}} \qquad (29)$$

From the above equation, DP^{Lat}denotes the processing latency in devices, TO^{La} denotes the overhead due to offloading, Se^{La} denotes the processing latency in servers, DTx^{La} denotes the latency during data transfer, and Ne^{La} is the network latency.

In general, latency must be as low as possible for the computational offloading tasks in cloud environment. We have shown the effectiveness of the proposed MFedL-MTD with the state-of-the-art works in fig 4 and table III. It is observed from the figure that, the latency increases with increase in task rate. Quantitatively, when we have minimum task rate of 5, the latency achieved by the MFedL-MTD gains 0.15s whereas the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO gains 0.25s, 0.39s, 0.45s, and 0.7s respectively. When the task rate is at maximum range of 25, the latency achieved by the MFedL-MTD is 1.15s whereas the existing works achieves 1.27s, 2.56s, 2.75s, and 2.92s respectively. The difference of latency when the task rate is minimum is (0.1s-0.8s), and when the task rate is maximum is (0.12s-1.77s) respectively.

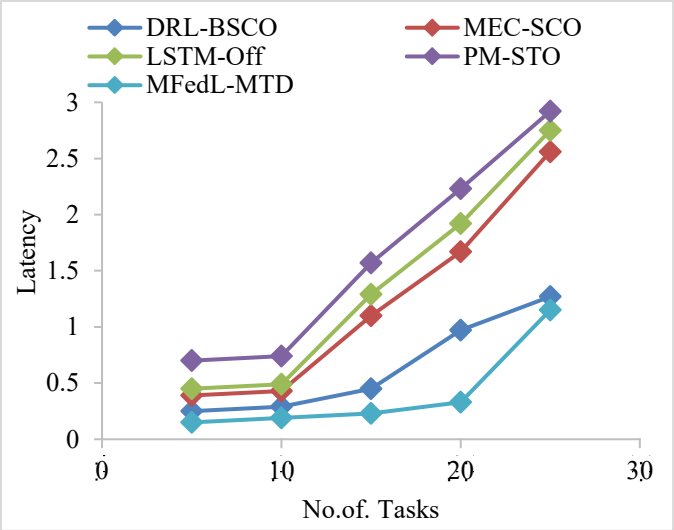


Fig 4. Performance Analysis of Latency

Table 3. Latency Performance Comparison

No. of Tasks	Latency (s)				
	DRL-BSCO	MEC-STO	LSTM-Off	PM-STO	MFedL-MTD
5	0.25s	0.39s	0.45s	0.7s	0.15s
10	0.29s	0.43s	0.49s	0.74s	0.19s
15	0.45s	1.10s	1.29s	1.57s	0.23s
20	0.97s	1.67s	1.92s	2.23s	0.33s
25	1.27s	2.56s	2.75s	2.92s	1.15s

(b) Computational Complexity Analysis

Computation complexity is defined as theassessment of the computational resources for task execution in the cloud environment. The mathematical formula for the computational complexity is provided below,

$$CC_{off} = \sum Com_{task} + Comm_{oh} + DM_{com} \qquad (30)$$

From the above equation, Com_{task}, $Comm_{oh}$, and DM_{com} denotes the complexity of task, communication overhead, and complexity due to decision making respectively.

The computational complexity also generally low as much as possible for the computational complexity task. In fig 5 and table IV, we have shown the graphical comparison of computational complexity of the proposed work with the existing works. From the graphical illustration, it is seen that the proposed MFedL-MTD achieves 2.5 when the offloading decision factor is at 2 whereas the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO achieves 3, 3.43, 3.72 and 3.85 respectively. The difference among the proposed and existing on minimum decision factor is (0.5-1.35). On the other hand, when the offloading decision factor reaches it maximum of 6, the computational complexity achieved by the proposed MFedL-MTD is 3.15 and for the

existing works are 8.5, 13.5, 20.35, 32.14 respectively. The maximum difference of computational complexity achieved when the offloading decision factor is at 6 is (5.35-28.99).

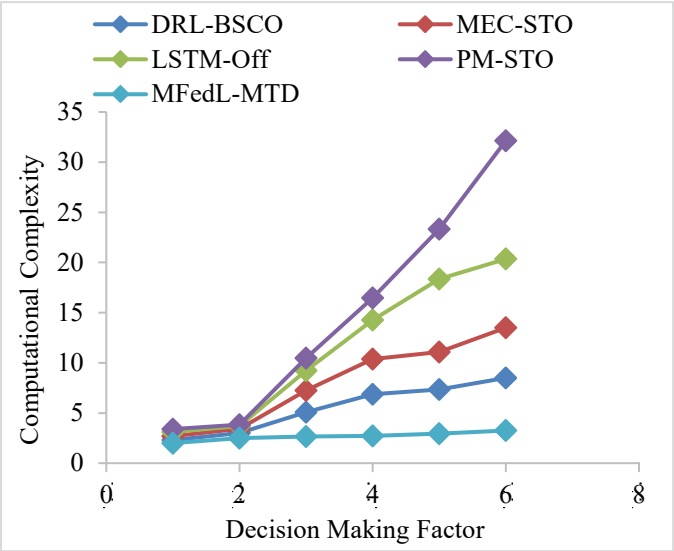


Fig 5. Performance Analysis of Computational Complexity

Table 4. Computational Complexity Performance Comparison

Decision Making Factor	Computational Complexity				
	DRL-BSCO	MEC-STO	LSTM-Off	PM-STO	MFedL-MTD
1	2.3	2.7	3.1	3.4	2
2	3	3.43	3.72	3.85	2.5
3	5.09	7.26	9.23	10.46	2.65
4	6.87	10.37	14.26	16.37	2.73
5	7.35	11.09	18.35	23.33	2.93
6	8.5	13.5	20.35	32.14	3.25

(c) Security Strength Analysis

The security strength refers to the assessment of incoming network forms in any forms (i.e. malicious users, external and internal network traffic respectively). The formulation of security strength is computed below as follows,

$$SS_{off} = \frac{\text{Malicious Network Traffic}}{\text{Overall Traffic}} \times 100 \qquad (31)$$

The above equation provides the ratio of malicious network traffic by ratioing the overall network traffic which also includes legitimate users into account.

The security strength analysis for the proposed MFedL-MTD and the existing works are illustrated in fig 6 and table V. From the figure, it is shown that when the number of malicious traffic rate increases, the security strength gets gradually decreases. Among that, the proposed MFedL-MTD model achieves higher security strength than the existing works. From quantitative analysis point of view, the proposed MFedL-MTD achieves higher security strength of 75% when the malicious traffic rate is at 10bits/sec whereas the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO achieves 69%, 63%, 58%, and 53% respectively. When the malicious traffic rate is reached to maximum of 100bits/sec, the proposed MFedL-MTD achieves 96% of security strength and for the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO gains 92%, 89%, 85% and 83% respectively. The difference when the minimum traffic rate among the proposed and existing work is (6%-12%), and for the maximum traffic rate is (4%-13%) respectively.

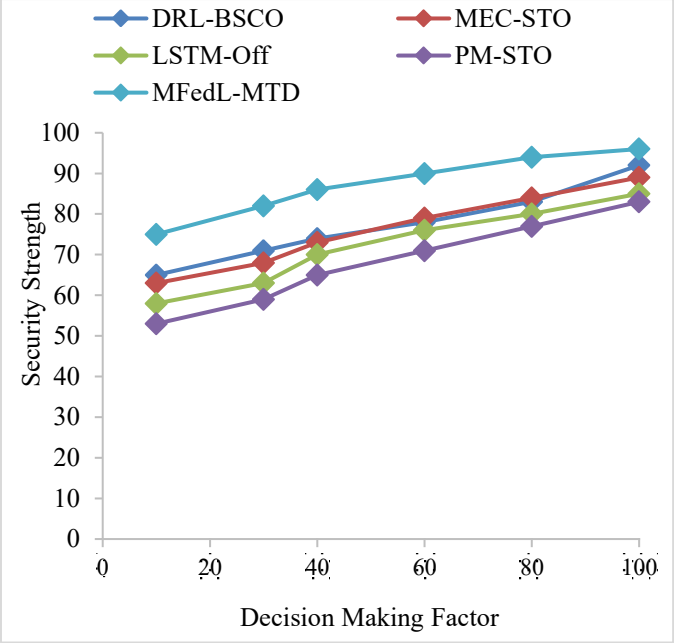


Fig 6. Performance Analysis of Security Strengthen

Table 5. Security Strength Performance Comparison

Decision Making Factor	Security Strength (%)				
	DRL-BSCO	MEC-STO	LSTM-Off	PM-STO	MFedL-MTD
10bits/sec	69%	63%	58%	53%	75%
30bits/sec	71%	68%	63%	59%	82%
40bits/sec	74%	73%	70%	65%	86%
60bits/sec	78%	79%	76%	71%	90%
80bits/sec	83%	84%	80%	77%	94%
100bits/sec	92%	89%	85%	83%	96%

(d) Migration Time Analysis

Migration time refers to computation of offloading time its takes in three levels. More clearly, it depends on various factors which includes size of data, bandwidth, and latency respectively. The formulation of migration time in cloud offloading tasks can be provided as follows,

$$\text{Migration Time} = \frac{\text{size}^{\text{data}}}{\text{BW}} + \text{Latency} \qquad (32)$$

From the above equation, $\text{size}^{\text{data}}$ denotes the data size to be migrated, and BW denotes the bandwidth of the network connected in bits/second.

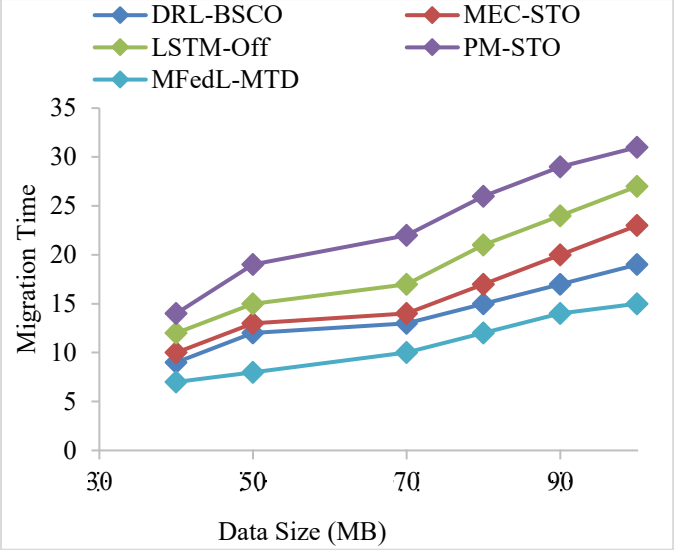


Fig 7. Performance Analysis of Migration Time

Fig 7represents the comparison of migration time assessment for the proposed and existing works with respect to average data size (MB). From the figure, we can see that the data size increases the migration time also increases. Among that, the proposed MFedL-MTD achieves lesser migration time than the existing works. More clearly, when the data size is in minimum size of 40MB, the migration time achieved by the proposed MFedL-MTD is 7mins and for the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO achieves 9mins, 10mins, 12mins, and 14 mins. The difference among the proposed and existing works when the data size gets minimum is (2mins-7mins). At higher size of data say it to 100MB, the proposed MFedL-MTD achieves nominal migration time of 15mins whereas the existing works achieves19mins, 23mins, 27mins, and 31mins respectively, and the difference achieved among the proposed and existing works are (4mins-16mins). Table VI provides the detailed representation of migration time analysis for the proposed and existing works.

Table 6. Migration Time Performance Comparison

Data size (MB)	**Migration Time (mins)**				
	DRL-BSCO	**MEC-STO**	**LSTM-Off**	**PM-STO**	**MFedL-MTD**
40MB	9mins	10mins	12mins	14mins	7mins
50MB	12mins	13mins	15mins	19mins	8mins
70MB	13mins	14mins	17mins	22mins	10mins
80MB	15mins	17mins	21mins	26mins	12mins
90MB	17mins	20mins	24mins	29mins	14mins
100MB	19mins	23mins	27mins	31mins	15mins

(e) Energy Consumption Analysis

Energy consumption refers to amount of energy consumed by the overall offloading environment. The energy consumption can be depending on the various factors such as mobile device energy consumption, transmission energy consumption, and cloud server energy consumption. The formulation for energy consumption is provided below,

$$\mathrm{ToT}^{\mathrm{EC}} = \mathrm{De}^{\mathrm{EC}} + \mathrm{Tx}^{\mathrm{EC}} + \mathrm{CS}^{\mathrm{EC}} \qquad (33)$$

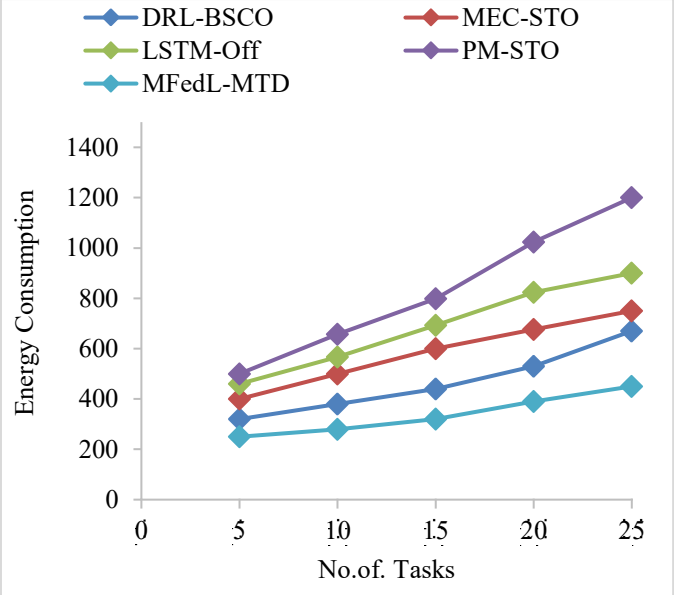


Fig 8. Performance Analysis of Energy Consumption

From the above equation, ToT^{EC} is the total energy consumption which can be summed up to De^{EC} device energy consumption, Tx^{EC} transmission energy consumption, and CS^{EC} cloud server energy consumption.

Fig8 and table VIIshows the energy consumption analysis for the proposed MFedL-MTD with the existing works with respect to number of tasks. It is illustrated in the figure that, when the task rate increases the energy consumption in the cloud environment also increases. Quantitatively, when the number of tasks is at minimum number 5, the energy consumption achieved by the proposed MFedL-MTD gains 250mJ whereas the existing works DRL-BSCO, MEC-STO, LSTM-Off, and PM-STO achieves 320mJ, 400mJ, 460mJ, and 500mJ respectively. When the number of tasks is at maximum of 25, the energy consumption achieved by the proposed MFedL-MTD is 450mJ and the existing works achieved 670mJ, 750mJ, 900mJ, and 1200mJ respectively. The difference between the proposed energy consumption when the energy consumption is at minimum range is (70mJ-250mJ) and for the maximum range the difference is (220mJ-750mJ).

Table 7. Energy Consumption Performance Comparison

No. of Tasks	**Energy Consumption (mJ)**				
	DRL-BSCO	**MEC-STO**	**LSTM-Off**	**PM-STO**	**MFedL-MTD**
5	320mJ	400mJ	460mJ	500mJ	250mJ
10	380mJ	500mJ	567mJ	657mJ	280mJ
15	440mJ	600mJ	693mJ	798mJ	320mJ
20	530mJ	676mJ	824mJ	1024mJ	390mJ
25	670mJ	750mJ	900mJ	1200mJ	450mJ

C. Federated Learning Analysis

In this section, we have compared the performance of the proposed multi-level federated learning secure aggregation (FL-SA) with the conventional federated learning averaging algorithms such as Federated Averaging (FedAvg), Federated Averaging with Momentum (FedAvgM), Federated Stochastic Variance Reduced Gradient (FedSVRG), Federated Optimal Transport (FOT). The proposed FL-SA performs optimal user selection by considering the user security constraints and task parameters (i.e. task type, task lifetime, etc..,). By considering those parameters, the proposed FL-SA performs optimal aggregation in multi levels without any noises (i.e. client noise ratio א, and ב client lower bound noise). Whereas the conventional models even though perform better in robustness, speed of convergence, and communication efficiency but lacks with client specific requirements in terms of security constraints. More clearly, the trade-off among the client requirements, communication security, and efficiency is lacked in the conventional FL aggregation settings. In fig 9 and table VII, we have shown the performance of proposed FL-SA with the validation metrics (accuracy, precision, recall, and F1-score) with respect to communication rounds on 8 devices. Quantitatively, we show the test accuracy (%) and standard deviation (%) on various FL settings and noise levels.

Table 8. Performance Evaluation of Proposed FL-SA with Conventional Federated Technique

FL Aggregation Method	Test Accuracy (%) ± Standard Deviation (%)			
	א = 0.0 ב = 0.0	א = 0.4 ב = 0.5	א = 0.6 ב = 0.5	א = 0.8 ב = 0.5
FedAvg	71.85±0.54	68.74±0.65	64.63±0.76	60.52±0.54
FedAvgM	67.26±0.36	63.23±0.47	60.15±0.58	57.65±0.69
FedSVRG	75.84±0.98	74.83±0.90	72.70±0.83	70.80±0.80
FOT	55.11±0.21	50.22±0.31	48.33±0.42	45.44±0.52
Proposed (FL-SA)	90.92±0.92	89.90±0.90	88.88±0.88	87.86±0.86

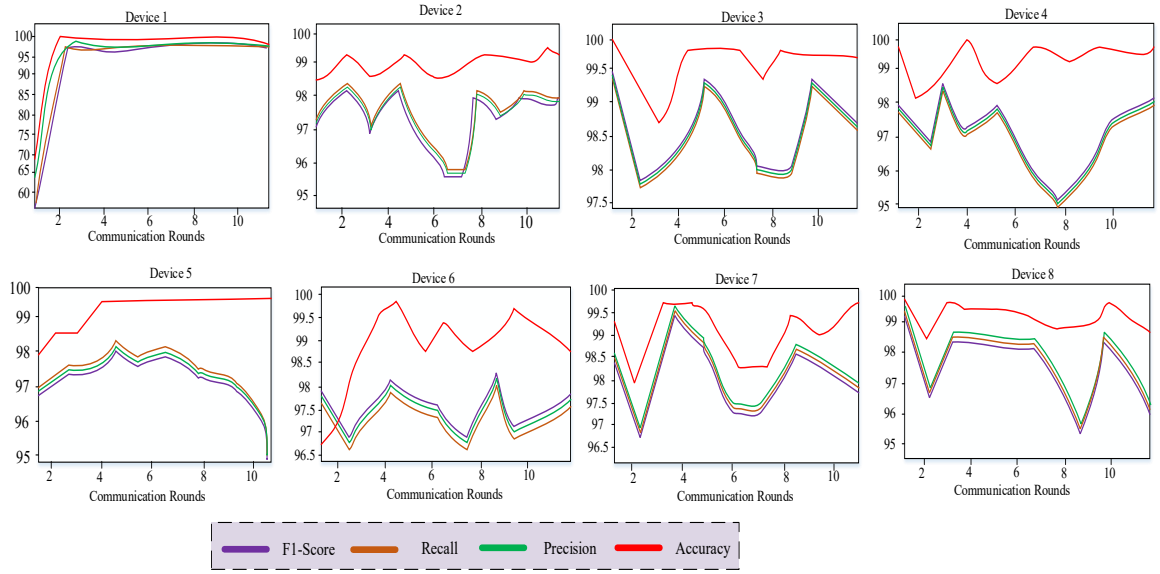


Fig 9. Comparative Analysis of Performance Metrics

Conclusion

In contemporary cloud security paradigms, optimal task offloading stands as a cornerstone, orchestrating the allocation of computational tasks to cloud resources in a manner that not only enhances efficiency and scalability but also fortifies resilience against potential security vulnerabilities. Task offloading in cloud computing is inherently accompanied by a spectrum of security concerns, spanning from safeguarding data privacy to mitigating the risks posed by unauthorized access and sensitive information leakage. Consequently, the imperative of optimal task offloading lies in its capacity to optimize resource utilization, curtail latency, and diminish attack surfaces, thereby significantly mitigating potential security risks and bolstering overall cloud security posture.While recent research endeavors predominantly focus on advancing task offloading methodologies, there remains a discernible gap in addressing critical aspects such as user privacy and the challenge of managing cloud server overflow scenarios. To bridge this gap and foster a more robust security framework, we introduce the MFedL-MTD framework, an innovative amalgamation of Multi-level Federated Learning with Moving Target Defense techniques. Within this framework, we institute Multi-factor Authentication to authenticate cloud users via a Trust Authority, ensuring the legitimacy of user interactions.

Furthermore, we leverage Multi-level Federated Learning for optimal user selection on edge servers, employing Federated Learning Secure Aggregation to enhance model aggregation performance. Task classification is facilitated using N-beats, while task scheduling incorporates relevant features to optimize performance. Notably, we integrate Moving Target Defense techniques for dynamic Virtual Machine (VM) placement and migration, particularly in instances of cloud server overflow, to engender an effective and secure task offloading environment.The efficacy of the proposed MFedL-MTD framework is assessed through a comprehensive evaluation encompassing various metrics such as latency, computation complexity, security strength, migration time, and energy consumption. Comparative analysis with state-of-the-art approaches serves to underscore the superior performance and efficacy of our framework in addressing contemporary challenges in cloud security and task offloading methodologies.

Reference

1. Mostafa, A.M., Ezz, M.M., Elbashir, M.K., Alruily, M., Hamouda, E., Alsarhani, M., & Said, W. (2023). Strengthening Cloud Security: An Innovative Multi-Factor Multi-Layer Authentication Framework for Cloud User Authentication. Applied Sciences.

2. Zheng, Y., Li, Z., Xu, X., & Zhao, Q. (2021). Dynamic defenses in cyber security: Techniques, methods and challenges. Digit. Commun. Networks, 8, 422-435.

3. Alghofaili, Y., Albattah, A., Alrajeh, N., Rassam, M.A., & Al-rimy, B.A. (2021). Secure Cloud Infrastructure: A Survey on Issues, Current Solutions, and Open Challenges. Applied Sciences.

4. Kaja, D.V., Fatima, Y., &Mailewa, A.B. (2022). Data Integrity Attacks in Cloud Computing: A Review of Identifying and Protecting Techniques. International Journal of Research Publication and Reviews.

5. Akello, P., Beebe, N. L., & Choo, K. K. R. (2022). A literature survey of security issues in Cloud, Fog, and Edge IT infrastructure. Electronic Commerce Research, 1-35.

6. Akhtar, N., Kerim, B., Perwej, Y., Tiwari, A., & Praveen, S. (2021). A Comprehensive Overview of Privacy and Data Security for Cloud Storage. International Journal of Scientific Research in Science Engineering and Technology.

7. Otta, S. P., Panda, S., Gupta, M., & Hota, C. (2023). A Systematic Survey of Multi-Factor Authentication for Cloud Infrastructure. Future Internet, 15(4), 146.

8. AlAhmad, A. S., Kahtan, H., Alzoubi, Y. I., Ali, O., & Jaradat, A. (2021). Mobile cloud computing models security issues: A systematic review. Journal of Network and Computer Applications, 190, 103152.

9. Liu, Z., Xu, B., Cheng, B., Hu, X., &Darbandi, M. (2022). Intrusion detection systems in the cloud computing: A comprehensive and deep literature review. Concurrency and Computation: Practice and Experience, 34(4), e6646.

10. Escaleira, P., Cunha, V. A., Gomes, D., Barraca, J. P., & Aguiar, R. L. (2023). Moving Target Defense for the cloud/edge Telco environments. Internet of Things, 24, 100916.

11. Jalowski, Ł., Zmuda, M., & Rawski, M. (2022). A Survey on Moving Target Defense for Networks: A Practical View. Electronics, 11(18), 2886.

12. Alouffi, B., Hasnain, M., Alharbi, A., Alosaimi, W., Alyami, H., & Ayaz, M. (2021). A systematic literature review on cloud computing security: threats and mitigation strategies. IEEE Access, 9, 57792-57807.

13. Katal, Avita, Susheela Dahiya, and Tanupriya Choudhury. "Energy efficiency in cloud computing data centers: a survey on software technologies." Cluster Computing 26.3 (2023): 1845-1875.

14. Sarkar, N., Keserwani, P. K., & Govil, M. C. (2023). A better and fast cloud intrusion detection system using improved squirrel search algorithm and modified deep belief network. Cluster Computing, 1-20.

15. Bayer, M., Frey, T., & Reuter, C. (2023). Multi-level fine-tuning, data augmentation, and few-shot learning for specialized cyber threat intelligence. Computers & Security, 134, 103430.

16. Vinoth, S., Vemula, H. L., Haralayya, B., Mamgain, P., Hasan, M. F., & Naved, M. (2022). Application of cloud computing in banking and e-commerce and related security threats. Materials Today: Proceedings, 51, 2172-2175.

17. Mostafa, A. M., Ezz, M., Elbashir, M. K., Alruily, M., Hamouda, E., Alsarhani, M., & Said, W. (2023). Strengthening Cloud Security: An Innovative Multi-Factor Multi-Layer Authentication Framework for Cloud User Authentication. Applied Sciences, 13(19), 10871.

18. Mostafa, A. M., Ezz, M., Elbashir, M. K., Alruily, M., Hamouda, E., Alsarhani, M., & Said, W. (2023). Strengthening Cloud Security: An Innovative Multi-Factor Multi-Layer Authentication Framework for Cloud User Authentication. Applied Sciences, 13(19), 10871.

19. Fu, X., Sun, Y., Wang, H., & Li, H. (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. Cluster Computing, 26(5), 2479-2488.

20. Flinn, J. (2022). Cyber foraging: Bridging mobile and cloud computing. Springer Nature

21. Rangwani, D., & Om, H. (2021). A secure user authentication protocol based on ECC for cloud computing environment. Arabian Journal for Science and Engineering, 46, 3865-3888.

22. Arasan, A., Sadaiyandi, R., Al-Turjman, F., Rajasekaran, A. S., & Selvi Karuppuswamy, K. (2021). Computationally efficient and secure anonymous authentication scheme for cloud users. Personal and Ubiquitous Computing, 1-11.

23. Lee, H., Kang, D., Lee, Y., & Won, D. (2021). Secure three-factor anonymous user authentication scheme for cloud computing environment. Wireless Communications and Mobile Computing, 2021, 1-20.

24. Al-Shqeerat, K. H., &Abuzanouneh, K. I. (2021). A hybrid graphical user authentication scheme in mobile cloud computing environments. International Journal of Communication Networks and Information Security, 13(1), 68-75.

25. Kaur, S., Kaur, G., & Shabaz, M. (2022). A secure two-factor authentication framework in cloud computing. Security and Communication Networks, 2022, 1-9.

26. Sucasas, V., Mantas, G., Papaioannou, M., & Rodriguez, J. (2021). Attribute-based pseudonymity for privacy-preserving authentication in cloud services. IEEE Transactions on Cloud Computing.

27. Vivekanandan, M., VN, S., & U, S. R. (2021). Blockchain based privacy preserving user authentication protocol for distributed mobile cloud environment. Peer-to-Peer Networking and Applications, 14, 1572-1595.

28. Wang, D., Zhao, N., Song, B., Lin, P., & Yu, F. R. (2021). Resource management for secure computation offloading in softwarized cyber–physical systems. IEEE Internet of Things Journal, 8(11), 9294-9304.

29. Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2021). Secure computation offloading in blockchain based IoT networks with deep reinforcement learning. IEEE Transactions on Network Science and Engineering, 8(4), 3192-3208.

30. Xu, S., Guo, C., Hu, R. Q., & Qian, Y. (2021). Blockchain-inspired secure computation offloading in a vehicular cloud network. IEEE Internet of Things Journal, 9(16), 14723-14740.

31. Samy, A., Elgendy, I. A., Yu, H., Zhang, W., & Zhang, H. (2022). Secure task offloading in blockchain-enabled mobile edge computing with deep reinforcement learning. IEEE Transactions on network and service management, 19(4), 4872-4887.

32. Arif, M., Ajesh, F., Shamsudheen, S., & Shahzad, M. (2022). Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing. Security And Communication Networks, 2022, 1-13.

33. Li, Z., Chang, V., Hu, H., Yu, D., Ge, J., & Huang, B. (2021). Profit maximization for security-aware task offloading in edge-cloud environment. Journal of Parallel and Distributed Computing, 157, 43-55.

34. Elgendy, I. A., Muthanna, A., Hammoudeh, M., Shaiba, H., Unal, D., & Khayyat, M. (2021). Advanced deep learning for resource allocation and security aware data offloading in industrial mobile edge computing. Big Data, 9(4), 265-278.

35. Saxena, D., Gupta, I., Kumar, J., Singh, A. K., & Wen, X. (2021). A secure and multiobjective virtual machine placement framework for cloud data center. IEEE Systems Journal, 16(2), 3163-3174.

36. Yadav, A. K., Bharti, R. K., & Raw, R. S. (2021). SA2-MCD: secured architecture for allocation of virtual machine in multitenant cloud databases. Big Data Research, 24, 100187.

37. Feizollahibarough, S., & Ashtiani, M. (2021). A security-aware virtual machine placement in the cloud using hesitant fuzzy decision-making processes. The Journal of Supercomputing, 77, 5606-5636.

38. Doyle, J., Golec, M., & Gill, S. S. (2022). Blockchainbus: A lightweight framework for secure virtual machine migration in cloud federations using blockchain. Security and Privacy, 5(2), e197.

39. Song, H., Li, J., & Li, H. (2021). A cloud secure storage mechanism based on data dispersion and encryption. IEEE Access, 9, 63745-63751.

40. Rafique, A., Van Landuyt, D., Beni, E. H., Lagaisse, B., & Joosen, W. (2021). CryptDICE: Distributed data protection system for secure cloud data storage and computation. Information Systems, 96, 101671.