Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A fog-edge-enabled intrusion detection system for smart grids

Noshina Tariq[1], Amjad Alsirhani[2], Mamoona Humayun[3*], Faeiz Alserhani[4] and Momina Shaheen[5]

Abstract

The Smart Grid (SG) heavily depends on the Advanced Metering Infrastructure (AMI) technology, which has shown its vulnerability to intrusions. To effectively monitor and raise alarms in response to anomalous activities, the Intrusion Detection System (IDS) plays a crucial role. However, existing intrusion detection models are typically trained on cloud servers, which exposes user data to significant privacy risks and extends the time required for intrusion detection. Training a high-quality IDS using Artificial Intelligence (AI) technologies on a single entity becomes particularly challenging when dealing with vast amounts of distributed data across the network. To address these concerns, this paper presents a novel approach: a fog-edge-enabled Support Vector Machine (SVM)-based federated learning (FL) IDS for SGs. FL is an AI technique for training Edge devices. In this system, only learning parameters are shared with the global model, ensuring the utmost data privacy while enabling collaborative learning to develop a high-quality IDS model. The test and validation results obtained from this proposed model demonstrate its superiority over existing methods, achieving an impressive percentage improvement of 4.17% accuracy, 13.19% recall, 9.63% precision, 13.19% F1 score when evaluated using the NSL-KDD dataset. Furthermore, the model performed exceptionally well on the CICIDS2017 dataset, with improved accuracy, precision, recall, and F1 scores reaching 6.03%, 6.03%, 7.57%, and 7.08%, respectively. This novel approach enhances intrusion detection accuracy and safeguards user data and privacy in SG systems, making it a significant advancement in the field.

Keywords Advanced metering infrastructure, Artificial intelligence, Edge computing, Federated learning, Fog computing, Intrusion detection system, Privacy, Smart grids, Support vector machine

Introduction

Smart grid (SG) infrastructures represent an advancement over conventional electricity grids with increased stability and efficacy to provide businesses and residences with uninterrupted power. It comprises communication and an energy network between consumers and power companies. The SG infrastructure depends on Advanced Metering Infrastructure (AMI) [1, 2], which comprises smart meters, edge devices, bidirectional communication connections, and a data aggregation cloud server [3], for collecting data, processing it, and employing control measures like remote appliance control in smart homes [4, 5]. However, as the volume of transferred data grows, challenges to the communication network also grow. Thankfully, 5G wireless communication technology constantly improves and offers fast transmission speed,

*Correspondence:
Mamoona Humayun
mahumayun@ju.edu.sa
[1] Department of Avionics Engineering, Air University, Islamabad 44000, Pakistan
[2] Department of Computer Science, College of Computer and Information Sciences, Jouf University, Al Jouf 72388, Saudi Arabia
[3] Department of Information Systems, College of Computer and Information Sciences, Jouf University, Al Jouf 72388, Saudi Arabia
[4] Department of Computer Engineering & Networks, College of Computer and Information Sciences, Jouf University, Al Jouf 72388, Saudi Arabia
[5] Department of Computing & School of Arts, Humanities and Social Sciences, University of Roehampton, London SW15 5PU, UK

Springer Open

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

an extensive bandwidth communication network, and little transmission delay [6, 7]. It makes SG and 5G amalgamation an expansion path in the future.

Deploying smart meters brought adaptable applications and diversified AMI services by integrating wireless technologies into the SG. However, The AMI system is vulnerable to attacks because of widespread distribution, unstable environment, and bidirectional communication networks [8]. The secrecy and availability of the AMI system may be harmed by attacks such as jamming [9], Man-In-The-Middle (MITM) [10], eavesdropping [11], False Data Injection (FDI), replay [12], and Denial of Service (DoS) [13] attacks. Figure 1 depicts the components of an SG and significant potential vulnerabilities. It represents the major SG components, such as smart meters, smart home appliances that use power, base stations, smart meter offices, and power sources. It further illustrates that intruders can exploit these vulnerabilities to disrupt SG functionalities. For instance, injecting false data into smart meters may jeopardize the accuracy of data analytics and reporting, resulting in manipulated power costs. Therefore, Intrusion Detection Systems (IDS) have been extensively researched to defend the AMI system's communication security. It can dynamically identify suspicious or anomalous behavior and promptly sound the alert [14–16]. To satisfy the criteria of AMI, it is crucial to create a compelling and quick detection IDS. The IDS based on Artificial Intelligence (AI) has been extensively implemented to increase the capacity to identify the IDS due to the development of AI. In these situations, the cloud server gathers much user data to build an intrusion detection AI-based model that is then utilized to watch over the AMI system on the cloud server side [17]. However, since personal information might directly disclose users' lifestyles, there is a risk that their privacy will be compromised, which could impact their lives[18]. Additionally, if the assault is close to the user's side, the detection latency would lengthen.

The majority of IDS models employ centralized cloud-based security mechanisms. The communication

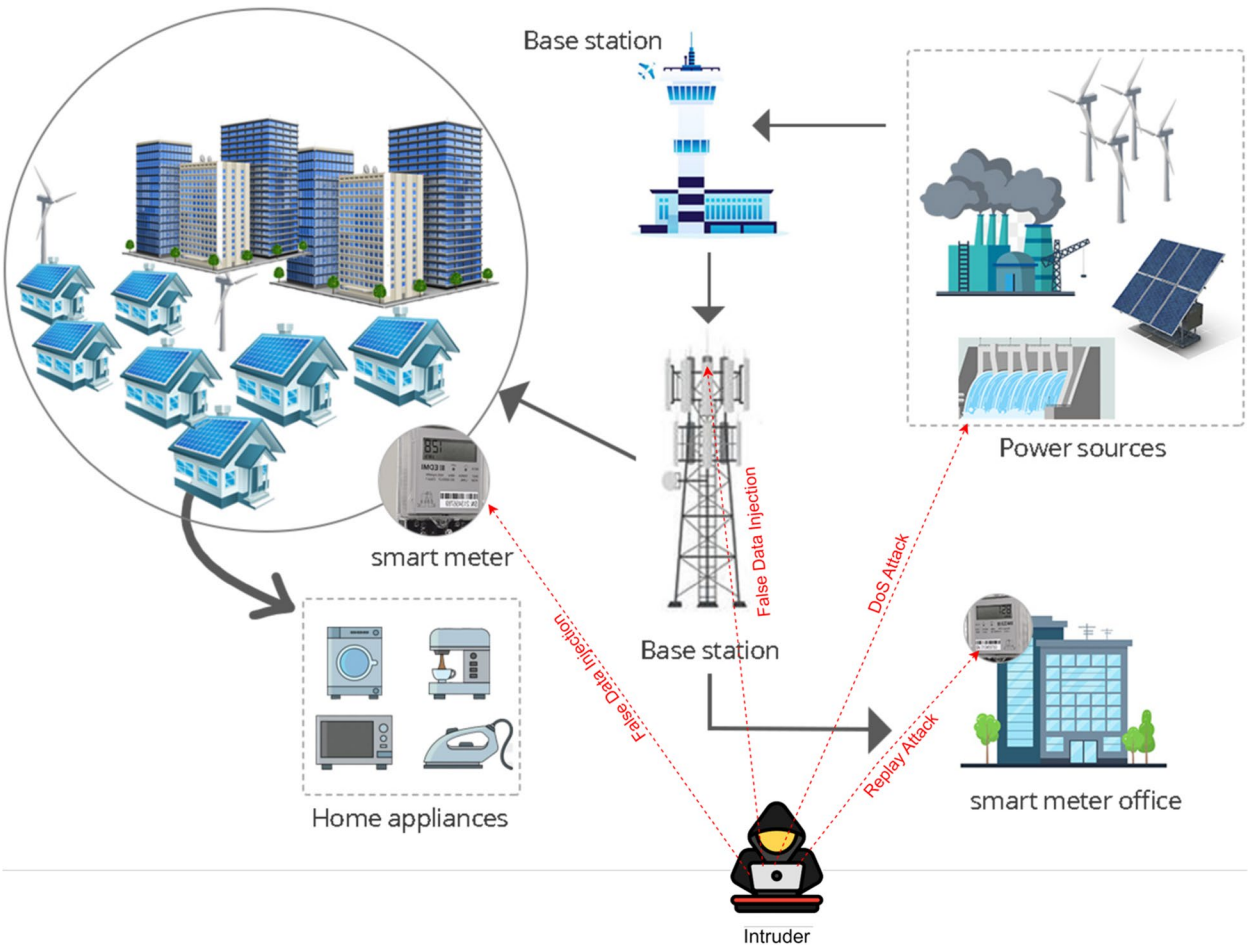

Fig. 1 SG Components and Potential Intrusion Vulnerabilities

constraints associated with centralized processing of the enormous amounts of data generated by interconnected devices include requirements for data transfer, battery life, memory usage, and delay [19]. SG networks, which are transient, are more susceptible to security risks due to their decentralized architecture. Their distributed nature necessitates a distributed security mechanism that supports scalability, interoperability, and adaptability with a uniform security system for heterogeneous connected devices. Therefore, fog computing offers distributed services for outsourcing computations of distributed SG architecture. Federated Learning (FL) comes to the rescue to solve the issue of data privacy. Multiple users train a shared model in tandem by sharing only the parameters with cloud servers in FL instead of raw data [20, 21]. The widespread use of such secure and distributed architecture encourages a privacy-preserved IDS for AMI. However, if the number of users increases, this feature might become a significant bottleneck as everyone uploads their local model to the cloud server in real-time, which may degrade network performance and increase resource utilization. Therefore, this paper presents an IDS based on FL with a Support Vector Machine (SVM) and a layered architecture with a fog layer. We preferred SVM due to its established efficacy in handling high-dimensional feature spaces and complex nonlinear decision boundaries. The rationale for this decision is to offer an innovative solution that efficiently identifies intrusions and tackles the increasing apprehensions regarding data privacy in cloud-based IDSs. Moreover, FL allows multiple users to collectively train a shared model without transmitting raw data. This secure and distributed architecture aligns seamlessly to establish a privacy-preserving IDS for AMI, ensuring user data remains secure and private. The proposed model offers improved intrusion detection performance, privacy preservation, real-time responsiveness, and scalability in SGs. It opens up new possibilities for secure and efficient IDSs in the era of edge computing and the Internet of Things (IoTs), where data privacy, latency, and resource constraints are critical considerations [22–24]. The key contributions of this paper are listed below:

1. A decentralized SVM-based collaborative model has been proposed using FL while preserving data privacy and handling high-dimensional feature spaces and nonlinear decision boundaries.
2. A distributed layered architecture is proposed using a fog-edge layer, unlike a conventional cloud, for efficient data processing, model training, and reducing latency. This layered approach enhances the scalability and responsiveness of the IDS, particularly in large-scale and resource-constrained environments.
3. Mathematical and formal presentation of various threat models, proof of theorems and propositions, and use cases are presented, articulating the characteristics and behaviors of different intrusion attacks for understanding and mitigating security threats in SG networks.
4. Benchmark evaluation and comparison are presented using established metrics such as accuracy, recall, precision, F1 score, and specificity between local and global, and global and state-of-the-art models.

The rest of the paper presents the background in "Background" section and related work in "Related work" section. Mathematical modeling and formal description are articulated in "Mathematical modeling and formal description" section. The proposed framework is detailed in "Proposed model" section. "Experimentation" section illustrates the experimentation details, and results are discussed in "Result analysis" section. Finally, a discussion is provided in "Discussion" section, and the conclusion is provided in "Conclusion" section.

Background

This section briefly discusses SG, intrusion detection in SG, FL, and SVM. Table 1 provides a glance at the topics discussed. The key topics (used in this paper) are SG, IDS, FL, SVM, and the significance of fog computing in SGs.

Smart grids

The SG, also known as the intelligent grid, is a notable technological progression within the energy system. The fundamental objective of conventional energy grids revolves around efficient electricity transmission from a centralized power generator to a substantial user base. In particular, the concept of the SG encompasses a paradigm shift from the traditional power grid, which relies on an electromechanical control system, to a digitally-driven and automated network featuring decentralized control capabilities [25]. This advanced grid model integrates various technologies, communication systems, and control mechanisms to optimize electricity generation, distribution, and consumption. The SG has six primary components: service provider, transmission, market, customer, operation, and distribution [26].

According to the research published in [40], the SG comprises many electronic-based sensing and communication technologies, control technologies, information/network management, and sensor field devices coordinating diverse electrical activities. The traditional grid architecture has undergone a significant transformation due to the integration of SG technologies and the emergence of operational challenges [27]. This transformation manifests in three crucial aspects: (a) enhancing the

Table 1 Summary of Background section

Topic	Description	References
SGs	The architecture and components of SGs, integrating diverse technologies and optimizing electricity generation.	Mohassel et al. [25], Gold et al. [26], Abou et al. [27]
Intrusion Detection	Types of IDS and their roles in cybersecurity for SGs.	Ahmad et al. [28], Khraisat et al. [29], Chandola et al. [30]
FL	Introduction to FL and its advantages in decentralized Machine Learning (ML).	Deepa et al. [31], Li et al. [32], Alazab et al. [33]
SVM	Explanation of SVM and their applications in image categorization, face recognition	Bansal et al. [34], Moqurrab et al. [35], Tanveer et al. [36]
Fog Computing in SGs	The significance of Fog Computing in SGs, enhancing data processing efficiency and security.	Hazra et al. [37], Singh et al. [38], Tariq et al. [39]

capacity to monitor and measure processes accurately, (b) transmitting relevant information back to management and control centers through an effective feedback mechanism, and (c) frequently responding automatically to adjust the system's response finely. This evolution in grid architecture has paved the way for increased efficiency and optimized performance [41]. Furthermore, it has facilitated the implementation of more refined control strategies for effectively managing electricity generation, distribution, and consumption. Second, coordinate data collection and transfer among various field devices and infrastructures. Finally, analyze, assess, and assist operators in accessing and using information derived by automated technology across the electrical grid [42]. Load prediction and balancing, grid reliability assessment, fault detection and monitoring, and grid security against cyber assaults are related issues in SGs [43–45].

The network architecture of the SG encompasses three interconnected domains: (a) the Home Area Network (HAN), (b) the Neighbourhood Area Network (NAN), and (c) the Wide Area Network (WAN). Each domain plays a distinct role in enabling the seamless functioning of the intelligent grid. Within this intricate framework, applications, such as Supervisory Control and Data Acquisition (SCADA), find their place within the expansive reach of the WAN, ensuring efficient monitoring and control of the grid on a broader scale [46]. On the other hand, the HAN and NAN encompass the essential components of the metering infrastructure, comprising smart meters and data concentrators. These localized networks empower households and neighborhoods to actively participate in the optimization of electricity consumption and contribute to the overarching intelligence of the grid [47]. Through the harmonious interplay of these interconnected networks, the SG paves the way for a more sustainable and technologically advanced energy ecosystem [48]. Considering the interdependence of these parties are attached to various forms of communication technology [49–51]. The system is increasingly vulnerable to internal and external assaults at higher network layers.

Intrusion detection

The primary function of an IDS is to keep an eye on all the information flowing across a network in search of any indications of malicious activity or unauthorized access. At this point, it either alerts a system administrator or takes preventative measures automatically [28]. There are typically two sorts of IDS. Threats are identified by anomaly-based IDSs when they deviate from the usual [29, 52]. On the other hand, signature-based IDS analyses incoming data for similarities to previously identified attack patterns and alerts administrators if any are found. IDS has vital roles in cybersecurity and can be classified into two distinct categories: host-based and network-based [53]. The host-based IDS consists of software applications strategically installed on individual client computers, diligently monitoring and safeguarding the integrity of their respective systems. On the other hand, the network-based IDS exhibits a broader reach, strategically positioned at multiple points within the network [28]. These hardware sensors or system software installations act as vigilant sentinels, meticulously examining the flow of data packets coursing through the network. By efficiently scrutinizing network traffic, these IDSs play a critical role in identifying and thwarting potential security breaches, ensuring the overall resilience and fortitude of the system [30]. The ideal deployment strategy for an SG may be either centralized or decentralized (or dispersed), depending on the nature of the environment [54].

When an SG is confronted with very real-time dynamic traffic patterns, conventional IDS are incapable of adequately handling the dynamic nature of the communication network. They must also be able to provide security services for already-in-use protocols, such as those that ensure privacy, availability, authenticity, and integrity. In addition, the IDS should accommodate system

maintenance cycles and limitations imposed by hardware and software [55]. The deployment of IDS in the SG presents unique challenges, primarily due to the profound implications of system security and the potential economic ramifications arising from attacks or malfunctions. The secure operation of the power system network is paramount, and this includes not only addressing network security constraints and ensuring the reliability of sensor networks and the intricate communication processes between utilities and consumers [56]. The sensor network is a critical system component in the SG. Consequently, addressing the security concerns associated with maintaining data integrity, availability, and secure connections throughout the network becomes imperative. Safeguarding these aspects is essential to mitigate risks and maintain the robustness of the SG ecosystem [57].

Federated learning

FL is a modern ML technique that shifts paradigms by decentralizing the global ML algorithm to individual devices rather than transmitting raw data from multiple sources to a central model [31]. In FL, the local parameters of each device are shared with the central ML algorithm, enabling global training and predictions (see Fig. 2). It represents that each client/local model uses its data and shares only the local parameters with the global aggregate model to preserve data privacy. The global model sends back the global model parameters after tuning. Diverging from conventional ML practices, which entail sending data to a central cloud server for training, FL distributes the models directly to the devices or locations where the data originates [32, 58, 59]. This approach offers several advantages, including enhanced privacy protection and efficient management of large datasets. Moreover, FL empowers decision-makers to harness real-time insights, facilitating prompt and informed decision-making [33].

It possesses a remarkable capability for collective intelligence by leveraging decentralized devices or servers, making it a key feature of this cutting-edge approach. Integrating AI techniques has emerged as a strategic choice with significant benefits in SG implementations. Modern electrical grids seamlessly incorporate distributed components of the SG ecosystem, including advanced communication frameworks, metering infrastructure, and distributed energy sources [60]. These components coexist with vast power networks and intricate communication systems, generating massive data. In such complex applications, the strategic utilization of AI techniques becomes imperative, serving as a cornerstone in effectively managing, analyzing, and extracting valuable insights from this data-rich landscape. By harnessing the power of AI, SGs are poised to achieve optimal performance, improved efficiency, and transformative advancements in energy management and delivery [61].

Support vector machine

SVMs were devised in 1963 by Alexey Ya. Chervonenkis and Vladimir N. Vapnik [34]. Since the introduction of SVM, this method has been extensively used to cope with several issues related to segmentation and categorizing images, hyperlinks, and texts. These algorithms exhibit a high level of sophistication and find applications in various domains, such as classifying proteins

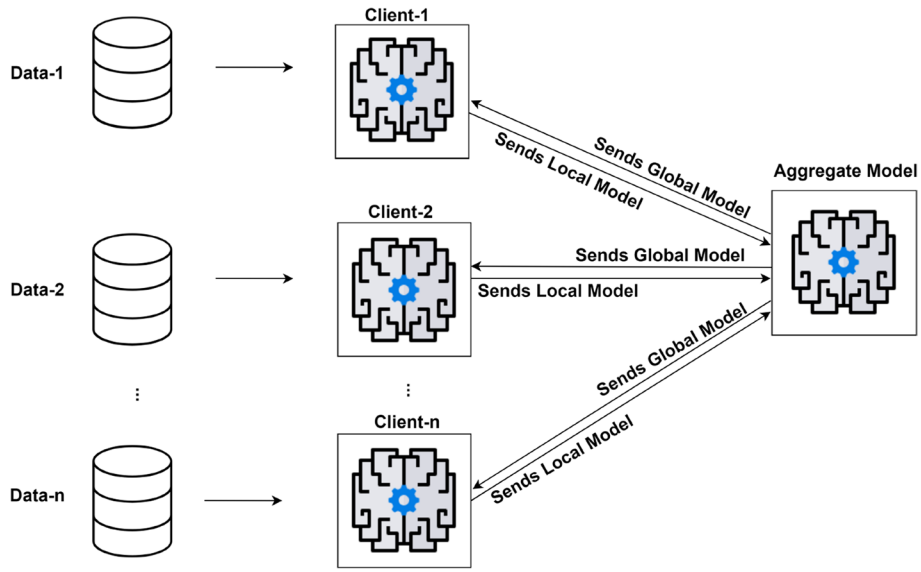

Fig. 2 Representation of an FL Model

in biological laboratories and recognizing handwritten text [35, 62, 63]. Additionally, they have proven to be instrumental in diverse fields, including autonomous vehicles, face recognition systems, and chatbots [64]. Among these algorithms, SVM is one of the most widely used supervised learning techniques for addressing regression and classification problems [34]. It employs the identification of support vectors, which are extreme data points, to facilitate the construction of an optimal hyperplane. This hyperplane is critical in effectively separating and classifying the data points based on their attributes [65].

Furthermore, the classification of images, text categorization, and Face detection are applications of the SVM algorithm. It takes data as input and performs initial classification on the given dataset. SVM classifiers produce weights or parameters that serve as input for the SVM classification (presented in Fig. 3). The classifier is initially trained using labeled data to create a decision boundary distinguishing between regular and intrusive instances. It dynamically updates the classifier by assigning weights to support vectors as new classification data is introduced. The SVM's ability to continuously adjust enables it to adapt to changing patterns and effectively classify network behavior. It makes it a reliable and adaptable solution for intrusion detection. For example, a person encounters an image of a peculiar cat that resembles a dog somehow. It helps develop a model that accurately distinguishes between a dog and a cat [36].

Fog computing in SGs

Fog computing, an innovative paradigm in distributed computing, emerges as a critical aspect of SG technology. It integrates seamlessly with the core framework of SG, providing a dynamic, flexible, and efficient computing platform that caters to the complex requirements of modern power systems. The basic premise of fog computing involves decentralizing the computational tasks traditionally handled by the cloud, bringing them closer to the data source [35, 37]. This architecture, which emphasizes edge computing, enhances data processing efficiency and reduces latency - aspects vital to ensure the smooth operation of SG [39]. Fog computing is inherited from cloud computing that moves data and processing closer to end users. It brings computation and data storage in proximity to the devices themselves rather than depending entirely on centralized cloud servers. This method mitigates two main issues with conventional cloud computing: (a) delays and (b) the requirement to send massive data across long distances [66]. It makes running context-aware, real-time applications and services possible even when there are gaps or interruptions to the network connection. This issue becomes more significant with the proliferation of internet-enabled devices and the consequent rise in data volume. Fog computing empowers organizations to make better, more timely choices through edge processing and analytics. Figure 4 illustrates a Fog-Cloud SG architecture. Fog at the network's edge enables decentralized data processing, reducing latency and improving real-time responsiveness. Meanwhile, the cloud enhances performance by offering scalable storage and advanced analytics capabilities, optimizing data processing, security, and reliability.

Fog computing is becoming more critical in advancing the development of SGs. These grids depend on real-time data from various sources, including sensors, smart

Fig. 3 Representation of SVM

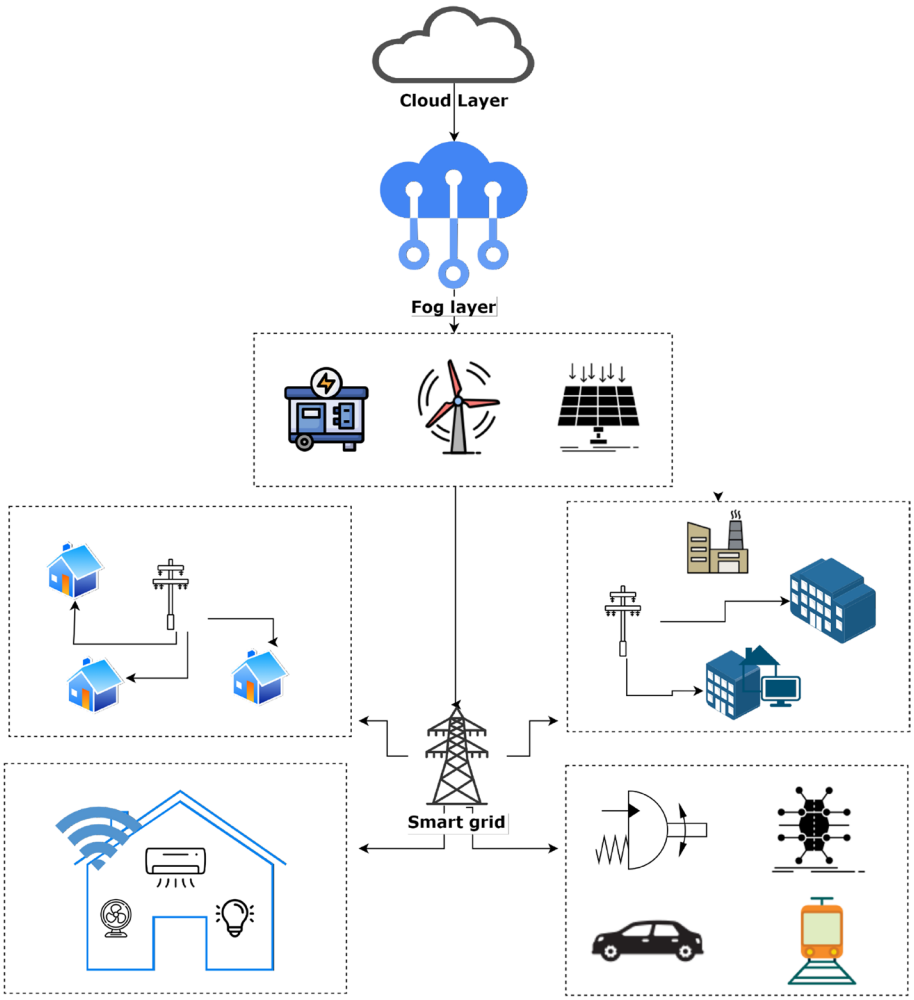

Fig. 4 SG based on Fog-Cloud computing

meters, and weather forecasts. SGs can improve energy use and reduce power outages by assessing this data at the network's edge. Rapid reaction time to supply and demand changes is essential to accomplish these objectives. In addition, SGs rely heavily on fog computing to keep private information secure [38, 67]. Fog computing aids data security by keeping data near its source and decreasing the need for long-distance data transfer. It is critical in SG installations, where data security is paramount. Fog computing is essential to realizing the aim of SGs to improve energy efficiency and dependability. Thanks to real-time data processing and secure edge computing, future energy networks may be more trustworthy and secure with the help of SGs. The fog layer, as described by NIST, is a transitional layer between endpoints (devices) and the cloud [68]. It offers storage and processing in real-time to handle the massive amounts of data produced by SG devices. It facilitates rapid data processing and decreases system latency by mediating communications between on-premises devices and remote cloud services. Integrating a fog layer into a SG system dramatically improves performance and reliability by supporting real-time processing and storage, eliminating latency, and bridging local devices and the cloud [69].

Related work

Several studies have targeted improving IDSs for SGs in the past few years. This section describes cutting-edge IDS techniques and methodologies.

Sagar et al. [70] proposed a fog-based IDS for SGs. They proposed a stacked model, using ensemble learning to represent the interdependencies among fog nodes susceptible to cyberattacks effectively. However, the dataset needs to be updated and reflect current attacks. Syed et al. [71] proposed an IDS for IoT that utilizes fog-cloud technology. This framework involves distributed processing by dividing the dataset based on attack class and a feature selection step

on time-series IoT data. The attack detection process involves utilizing deep learning Recurrent Neural Network models. The proposed approach was assessed using the high-dimensional BoT-IoT dataset, including realistic IoT attacks. As previously noted, reliability in the SG's electricity and services depends on the AMI system's security. Because of this, a safe SG must incorporate intrusion detection techniques within its AMI systems. As AI has progressed, several new ML-based techniques have been used in intrusion detection for AMI systems. To ensure the safety of the AMI, a novel IDS was proposed in [72] using a transformer and feature extraction layers to analyze categorical and numerical information. Another IDS was developed for Modbus/TCP and DNP3-based SG systems in [73]. It uses an Autoencoder-Generative Adversarial Network (GAN) architecture to identify irregular operations and categorize cyberattacks. Compared to other ML and DL approaches, the assessment findings show that it is the most effective due to its low false positive rate, high accuracy, and high true positive rate. Real-time intrusion detection was proposed in [74] using a Deep Neural Network (DNN) model hosted on a web server. Since deep-learning-based algorithms only accept numerical features as input, the one-hot encoding method is typically used to transform the categorical characteristics of a data sample into numerical features. However, this results in a high-dimensional and sparse input characteristics vector, which degrades intrusion detection capabilities. While the above techniques are decentralized, they rely on a central server for tasks like model training and intrusion detection. Users' (smart meters') locally gathered data is easily accessible by the cloud server under the centralized architecture, which raises privacy concerns. The centralized architecture additionally exacerbates the latency in detecting an assault near the user.

Another revolutionary FL-based methodology was proposed in [75] for detecting cyberattacks in SGs. It paves the way for cooperative attack detection model training without disclosing sensitive information about available power sources. The system uses a gradient privacy-preserving quantization approach to increase communication efficiency and a Deep Auto-Encoder network for precise anomaly identification. The simulation results show higher detection accuracy and communication efficiency. Similarly, an alternative FL model for detecting intrusion in SG AMI networks was proposed in [76]. To get the best results from the model, they used fine-tuned DNN. Multiple data concentrators can work together to improve the intrusion detection model through shared learning and improved sensitivity. In addition, the IDS model is set up on the data concentrators themselves, allowing for continuous monitoring and protection of sensitive information. To identify intrusions in a network, a multi-stage CLAIRE was proposed in [77], using nearest neighbor-based search, clustering, and Convolutional Neural Networks (CNNs). It converts network flows' one-dimensional feature vector into a two-dimensional image representation. Compared to other deep learning and clustering-based systems, CLAIRE's accuracy and intrusion detection performance fare better. This study integrates SVM, game theory, and kernel functions. The researchers in [78] suggested an FL-based IDS to deal with problems like the absence of authentication and encryption. The FL method protects sensitive information across the network while detecting intrusions. IT successfully identifies significant threats on smart meters since the model can be trained without exposing sensitive private data. An alternative FL-based IDS is proposed in [79] without needing a central training data repository, thereby eliminating human interaction. The system can identify any suspicious variations in the way devices communicate with one another. The goal is to study the invaders' behavior patterns while actively attempting to compromise a system's services. The updated system model is transmitted to a central server to identify patterns of assault.

A comparison of several intrusion detection strategies for SGs is presented in Table 2. The methods mentioned above encompass a variety of approaches, including DL-based approaches, FL, and other ML models. Every technique presents distinct positive and negative aspects, crucial factors to consider when choosing a suitable IDS for SGs. The existing literature offers valuable insights into different intrusion detection techniques for SGs. However, most existing approaches, such as [72, 73, 76], provide a centralized solution. In contrast, this paper caters to the issues associated with centralization and proposes a decentralized approach. In addition, this paper offers a decentralized SVM-based collaborative model that utilizes FL to address the challenges of preserving data privacy and handling high-dimensional feature spaces and nonlinear decision boundaries, unlike [74]. In addition, utilizing a distributed layered architecture with a fog-edge layer instead of a conventional cloud represents a deviation from the conventional approaches. The authentication, encryption, and convergence issues are effectively resolved in the proposed model compared to [77, 78]. This strategic shift aims to improve scalability, responsiveness, data processing, model training, and latency reduction. These factors are essential for achieving optimal IDS performance, particularly in large-scale and resource-constrained environments.

Table 2 Comparison of IDS Techniques for SGs

Ref.	Technique	Advantages	Disadvantages
[72]	Transformer and feature extraction layers	High accuracy, low false positives	Centralized architecture, data dependency
[73]	Autoencoder-GAN architecture	Low false positive rate, high accuracy, high true positive rate	Centralized architecture, model complexity
[74]	DNN	Real-time detection, feature learning	High-dimensional input, overfitting
[75]	FL with gradient privacy-preserving quantization	Higher detection accuracy, privacy-preserving	Communication efficiency, convergence
[76]	Fine-tuned DNN in FL	Improved sensitivity, continuous monitoring	Centralized architecture, scalability
[77]	CLAIRE with CNNs	Improved accuracy and performance	Complex model, training data requirements
[78]	FL-based IDS	Protects sensitive data, collaborative training	Lack of authentication and encryption, convergence
[79]	FL-based IDS	No central data repository, behavior pattern identification	Privacy-preserving, communication overhead

Mathematical modeling and formal description

This section details the rigorous mathematical framework, including threat models and formal description, to understand intrusion and its detection in SGs. It comprehensively explains the underlying principles and mechanisms that drive the proposed model. In addition, it examines different use cases to illustrate the practical implementations of the proposed methodology. Table 3 describes all the symbols used in subsequent sections.

Threat model

1: Compromised Smart Meter Leading to Disruption in Power Distribution

1. Assumptions: We assume an SG infrastructure consisting of multiple smart meters μ with a subset as X where $X = x_1, x_2, \ldots, x_n$. These smart meters are interconnected within the SG network, enabling data collection and power distribution management.
2. Attack Model 1: Attack on power distribution: The attack Model involving a compromised X and its impact on power distribution can be described as follows:
   (a) Unauthorized Control: An attacker specifically targets X and gains unauthorized control over it, denoted as $C(X)$, through various means, such as exploiting firmware vulnerabilities or compromising the communication protocol.
   (b) Data Manipulation: With unauthorized control, the attacker manipulates the energy consumption readings of X, resulting in altered or fabricated data, denoted as $M_{Data}(X)$.
   (c) Ripple Effect: The manipulated data, $M_{Data}(X)$, has a ripple effect on the SG infrastructure. It propagates to the central data hub, denoted as D_{Hub}, which processes data from all smart meters for power distribution management.
   (d) Power Disruption: Due to the manipulated data, D_{Hub} makes incorrect decisions regarding power distribution. It can lead to imbalances, fluctuations, or even disruptions in the power distribution across the residential area served by the SG. The resulting disruptions are denoted as P_{DS}.
   (e) Financial Losses: The manipulated energy consumption data also affects billing accuracy. Inaccurate billing can lead to financial losses for the utility company, denoted as F_{LS}. Customers may dispute overcharged bills, and the utility company incurs additional expenses for investigating and resolving billing discrepancies.
3. Remediation and Investigation: Upon detecting abnormalities in energy consumption patterns or identifying billing discrepancies, the utility company's security team initiates an investigation, denoted as I_{IV}. It aims to identify the compromised smart Meter, Meter X, and determine the attack's extent. Appropriate remediation measures are then taken to secure and restore the regular operation of Meter X, such as IDS.

Threat model

2: Firmware Exploitation We consider exploiting firmware vulnerabilities in smart meters within the SG network in this threat Model. Exploitation of firmware vulnerabilities poses significant risks to the integrity and reliability of the SG network. The following steps outline the phases and actions involved:

Table 3 Table of Symbols Used

Symbol	Description
CT_i	Each Client in the FL setup, denoted as CT_i
L_{SVMi}	Local SVM model associated with each client, CT_i
l_{mp_i}	Local model parameters of SVM model CT_i, including weights and biases
G_{svm}	Global SVM model, representing the collective knowledge of all clients in the FL setup
G_{mp}	Global model parameters, including the weights and biases of the global SVM model
Fed_{SVM}	Function representing the FL process with SVM
Ud_{SVM}	Function used for updating SVM weights in the FL process
K	Number of smart meters in the FL setup
T	Number of iterations in the FL process
l_w	Initial SVM weights
C	Regularization parameter used in SVM training
l_b	Initial SVM bias
w_g	Global SVM weights
b_g	Global SVM bias
μ	Smart Meters
X	Subset of Smart Meters
V	Vulnerability
$V = \{v_1, v_2, \ldots, v_n\}$	Set of Vulnerable Models
v_i	Vulnerable Smart Meter Model
F	Firmware of Smart Meters
F_{mal}	Malicious Firmware Update
$X = \{x_1, x_2, \ldots, x_n\}$	Set of Smart Meters
$X = \{x_1, x_2, \ldots, x_m\}$	Set of Compromised Smart Meters
$F(X) = F_{mal}$	Installation of Malicious Firmware
$R(X) = R_{man}$	Manipulation of Meter Readings
$C(X) = C_{man}$	Tampering with Consumption Data
$D(X) = D_{ds}$	Disruption of Communication
$G(N \setminus X) = G_{com}$	Compromise of Other Devices
$C(X)$	Unauthorized Control
$M_{Data}(X)$	Manipulated Energy Consumption Data
D_{Hub}	Central Data Hub
P_{DS}	Power Disruption
F_{LS}	Financial Losses
I_{IV}	Investigation Phase
α	Set of Attackers
P	Set of Potential Vulnerabilities
$T \subseteq \mu$	Target Smart Meters
$P(T) = P_{idf}$	Identified Vulnerabilities
G	Compromised Network Devices
$G(N) = G_{com}$	Compromised Devices
$X \subseteq \mu$	Compromised Smart Meters
N_d	Set of Network Devices (Gateways, Routers)

1. Intrusion Phase: During the intrusion phase, the attacker identifies a specific smart meter model with known firmware vulnerabilities. We represent the set of smart meters as μ and the set of attackers as α. Additionally, we denote the set of smart meter models with known firmware vulnerabilities as V, where $V = v_1, v_2, ..., v_n$. The attacker selects a vulnerable smart meter model $v_i \in V$ as their target for exploitation.
2. Exploitation Phase: Once the vulnerable smart meter model is identified, the attacker proceeds to exploitation. Let F represent the firmware of the μ. The attacker crafts a malicious firmware update or code injection payload denoted as F_{mal}. This payload is designed to exploit the specific firmware vulnerabilities in the target smart meter model.
3. Harmful Actions: The successful exploitation of firmware vulnerabilities enables the attacker to perform various harmful actions on the compromised smart meters. Let μ be the smart meters and $X \subseteq \mu$ be the set of compromised smart meters: $X = x_1, x_2, ..., x_m$. The attacker can perform the following actions on the compromised μ:
   (a) Installation of Malicious Firmware: The attacker successfully installs the malicious firmware on the compromised μ, replacing the legitimate firmware. This is represented as $F(X) = F_{mal}$.
   (b) Manipulation of Meter Readings: The attacker gains the ability to manipulate the meter readings of the compromised μ. The readings can be altered to reflect inaccurate or falsified energy consumption data. This is denoted as $R(X) = R_{man}$.
   (c) Tampering with Consumption Data: The attacker can tamper with the consumption data reported by the compromised μ. It includes modifying the recorded energy consumption values or injecting false data. The manipulated consumption data is denoted as $C(X) = C_{man}$.
   (d) Disruption of Communication: The attacker can disrupt the compromised μ communication channels, hindering their ability to send or receive data. This disruption is denoted as $D(X) = D_{ds}$.

Threat model

3: Network Exploitation In this threat Model, we examine the exploitation of network vulnerabilities within the SG infrastructure. These network vulnerabilities

pose significant risks to the network's confidentiality, integrity, and availability. Let μ represent the set of smart meters, N_d represent the set of network devices (communication gateways, routers), and α represent the set of attackers. The following steps outline the phases and actions involved:

1. Reconnaissance Phase: During reconnaissance, the attacker systematically assesses the SG network to identify potential vulnerabilities. We represent the set of smart meters as μ, the set of network devices (communication gateways, routers) as N_d, and the set of attackers as α. Additionally, we denote the set of potential vulnerabilities in the network infrastructure as P. The attacker's objective is to identify a specific target smart meter, denoted as $T \subseteq \mu$, and potential vulnerabilities associated with these meters, represented as $P(T) = P_{idf}$.
2. Unauthorized Access: Once the potential vulnerabilities are identified, the attacker gains unauthorized access to the targeted T. We denote the set of compromised network devices as G. The attacker successfully compromises a subset of network devices, specifically those associated with the μ of interest. This is represented as $G(N) = G_{com}$.
3. Harmful Actions: Upon compromising the network devices and gaining access to T, the attacker can carry out various harmful actions. We consider the set of compromised smart meters as $X \subseteq \mu$. The attacker can perform the following actions on the compromised X:
    (a) Manipulation of Meter Readings: The attacker can manipulate the meter readings of the compromised X. This manipulation can result in inaccurate or falsified energy consumption data. It is represented as $R(X) = R_{man}$.
    (b) Injection of False Control Commands: The attacker can inject false control commands into the compromised X, leading to unintended and potentially harmful actions. This injection of false commands is denoted as $I(X) = I_{inj}$.
    (c) Exfiltration of Data: The attacker can exfiltrate sensitive data from compromised X, such as customer information or energy consumption patterns. This unauthorized data exfiltration is represented as $E(X) = E_{exf}$.
    (d) Compromise of Other Network Devices: The compromised X can be a foothold for the attacker to compromise other network devices within the SG infrastructure. This is denoted as $G(N \setminus X) = G_{com}$.

Case study 1: securing SGs against intrusion attacks

1. Background: The case study focuses on an SG deployment in a metropolitan area consisting of thousands of smart meters and a central control system managed by a utility company. Smart meters enable automated meter reading and facilitate efficient energy management. However, with the increasing reliance on SG technologies, the risk of intrusion attacks has become a significant concern.
2. Definition: Intrusion in SG Smart Metering: An intrusion in SG smart metering is an unauthorized and malicious activity or event that compromises the security and functionality of smart meters within the SG infrastructure. Let μ represent the set of smart meters in an SG deployment, and α represent the set of attackers attempting to intrude on the SG smart metering system. An intrusion attack on μ can be modeled as follows:
    (a) Intrusion Phase: Let $X \subseteq \mu$ be the set of compromised smart meters: $X = x_1, x_2, ..., x_n$. The attacker gains unauthorized access to X in μ through various means, such as exploiting vulnerabilities, unauthorized physical access, or compromising network communication channels, as mentioned above.
    (b) Consequences: The consequences of intrusion in SG smart metering can include financial losses, compromised grid operations and energy management, privacy breaches, and safety hazards or equipment damage due to improper control settings, as mentioned above.
    (c) Analysis and Findings: Upon analyzing the current state of IDS in SG smart metering, several key findings emerge:
        i. Limited Local Detection Capabilities: Existing IDS solutions deployed at individual smart meters (AMIs) often have limited computational resources and lack sophisticated detection algorithms. It restricts their ability to detect and mitigate complex intrusion attacks effectively.
        ii. Data Privacy Concerns: Traditional IDS architectures rely on centralized data collection and analysis, which raises concerns about data privacy and the secure transmission of sensitive information from smart meters to a central control system.

iii. Scalability Challenges: As the number of smart meters in an SG deployment increases, the centralized IDS approach faces scalability challenges due to the growing volume of data to be processed and the potential for increased false positives.

(d) Recommendations: To address the identified challenges and enhance the security of SG smart metering systems, the following recommendations are proposed:

i. IDS with FL: Implement an IDS architecture that leverages FL techniques. This approach allows training a global IDS model using data distributed across multiple smart meters while preserving data privacy. Local models residing at the AMIs can perform initial detection, and only aggregated model updates are shared with a fog-based global model residing in the fog layer for further analysis and refinement.
ii. Fog Computing for Enhanced Processing: Utilize fog computing infrastructure located closer to the smart meters to enable real-time processing and analysis of IDS data. Moving some computation tasks closer to the edge can reduce latency, enabling faster detection and response to intrusion attacks.
iii. Secure Communication Channels: Establish secure communication channels between smart meters and the fog-based global IDS model. Encryption and authentication mechanisms should be implemented to ensure the confidentiality and integrity of data transmitted between the smart meters and the IDS infrastructure.
iv. Continuous Model Improvement: Implement a mechanism for continuous model improvement through regular model updates and feedback loops. The global IDS model in the fog layer should be regularly updated with the latest detection algorithms and threat intelligence to enhance its detection capabilities.

By adopting the recommended approach of IDS with FL, leveraging fog computing, and ensuring secure communication channels, SG smart metering systems can benefit from improved intrusion detection capabilities, enhanced data privacy, and efficient processing at the edge. These measures will contribute to the overall security and resilience of SGs against intrusion attacks.

Proposed model

The suggested architecture aims to increase the security of SGs against intrusion assaults by utilizing a decentralized IDS. This part includes a case study to help clarify the proposed model better and an introduction to the suggested architecture, which combines fog computing with FL to detect intrusions efficiently and privacy-preserving. The design attempts to achieve scalability, real-time detection, and resistance against intrusion attacks by spreading IDS functionality across smart meters and fog nodes. The architecture facilitates information aggregation from numerous smart meters while maintaining data privacy due to the collaborative nature of FL. The global fog layer acts as a coordination and aggregation point rather than a single control point. The decentralization aspect is introduced by distributing local models and training processes across AMIs and fog nodes. The significant characteristics of the proposed model are as follows:

1. Distribution of Local Models: In the decentralized IDS architecture, each fog node hosts a global model that performs intrusion detection on the parameters collected from the smart meters (running local models) within its vicinity. These local models are trained using SVM-enabled FL, allowing the models to learn from the data without needing centralized data aggregation. The local models are trained collaboratively, with updates and improvements shared between them through the global server.
2. FL and Collaboration: FL enables the collaborative training of local models without compromising data privacy. Instead of transmitting raw data to the fog layer, the local models send model updates or parameters to the global server, aggregating and combining the updates from all participating nodes. This approach ensures that sensitive data remains on the local nodes, preserving privacy and reducing the risks associated with centralized data storage.
3. Coordination and Aggregation: The training and aggregating model updates from the local models (i.e., the AMIs) are done by the global server (i.e., the fog). It collects the updates using privacy-protecting methods, such as secure aggregation or differential privacy, to ensure that each node's contributions stay private. It is then shared with the local models so that they can learn from their collective experience while still keeping their characteristics.
4. Decentralization: The decentralized architecture of the IDS has various benefits. It divides the processing load among several Fog nodes, enhancing scalability and lessening dependency on a single central server. It enhances data privacy by controlling sensi-

tive information and minimizing data transmission. In addition, the collaborative nature of FL enables knowledge sharing and model improvement among the local models, resulting in enhanced detection accuracy and adaptability.

Case study 2: a decentralized IDS system for SGs

The case study focuses on enhancing the security of SGs against intrusion attacks by implementing a decentralized IDS. Traditional IDS solutions face challenges in scalability, privacy, and real-time analysis. Therefore, a decentralized approach using fog computing and FL is proposed to address these limitations.

IDS architecture

The decentralized IDS architecture consists of two key components:

1. Smart Meters (SM): These are the metering devices deployed at consumer premises that collect energy consumption data and transmit it to the Fog layer. Each smart meter runs a local IDS model to detect anomalies and potential intrusions.
2. Fog Layer: The Fog layer is an intermediate computing platform between the smart meters and the central control system. It comprises Fog nodes, which are geographically distributed and closer to the smart meters. Each Fog node hosts a local IDS model and coordinates the training process with other nodes in an FL framework. It is responsible for aggregating and analyzing the information collected from the SMs. It maintains a global IDS model that combines the knowledge learned from the local models to make accurate intrusion detection decisions.

Decentralized IDS workflow

The decentralized IDS system follows the following workflow:

1. Local Model Training: Each smart meter uses locally collected data to train its local IDS model. This training phase incorporates SVM-based algorithms suitable for handling high-dimensional feature spaces and nonlinear patterns in intrusion detection.
2. FL: The fog nodes collaborate in an FL framework to improve the IDS models collectively. Only model parameters are exchanged instead of sharing raw data to preserve data privacy. Each fog node contributes its local model updates to the aggregator module, where the global IDS model is refined using federated averaging.
3. Model Distribution: The updated global IDS model is distributed back to the local nodes, ensuring the latest knowledge is disseminated across the decentralized IDS system. They then update their local models with the refined global model, enabling them to adapt to evolving intrusion patterns.
4. Real-time Intrusion Detection: Each smart meter utilizes its updated local IDS model for real-time intrusion detection. Anomalies and potential intrusions are detected by comparing the observed metering data with the learned patterns in the local model. Alerts are generated when suspicious activities are identified.

Advantages of decentralized IDS

The deployment of a decentralized IDS system using a fog-enabled FL model offers several advantages:

- Scalability: The decentralized architecture allows for seamless scalability as new smart meters can join the system without requiring extensive modifications. The FL approach ensures that the IDS system can benefit from the collective knowledge of a large number of participating smart meters.
- Privacy Preservation: By leveraging FL, the IDS system preserves data privacy. Raw metering data is kept locally, and only aggregated model updates are shared across the network. It protects sensitive consumer information while allowing the system to learn from diverse data sources.
- Real-time Detection: With the IDS models deployed at the AMIs and fog layer, real-time intrusion detection can be achieved. The proximity of fog nodes to smart meters enables faster analysis and response, reducing the detection latency compared to a centralized approach.
- Resilience: The decentralized IDS system enhances system resilience by avoiding a single point of failure. Even if some fog nodes or smart meters become compromised, the collective intelligence of the distributed IDS models can continue to provide effective intrusion detection and mitigation.

FL-based IDS

In this section, the proposed mechanism for the detection of intrusion for SG systems is explained. The key characteristics are as follows:

1. Each Client's Local SVM: Each Client (CT_i) in the FL setup has its own local SVM model ($L_{SVM\,i}$). The local SVM is trained using the Client's local dataset, which consists of data samples not shared with other clients or the central server. The local SVM learns the patterns and characteristics of the data specific to each Client, allowing it to make predictions locally.
2. Local Model Parameters: After training the local SVM, the Client obtains the local model parameters (i.e., l_{mp_i}) associated with its SVM model. These parameters represent the learned weights and biases of the SVM, which define the decision boundary and enable it to classify new data samples.
3. Communication with the Server: The local model parameters are sent from each Client to the central server. This communication can be done securely, ensuring the privacy of the local data and model.
4. Global SVM and Global Model Parameters: At the server, the received local model parameters are aggregated and used to update the (G_{svm}). It represents the collective knowledge of all clients and aims to capture a generalized representation of the data. The server computes the global model parameters (G_{mp} based on the aggregated local model parameters.
5. Model Parameter Exchange and Iterative Training: The updated global model parameters are returned to the clients for further training and model improvement. This parameter exchange and iterative training process allow the clients to benefit from the collective knowledge of the entire network while preserving the privacy of their local data. The iterative training continues until the global SVM converges to a desired level of accuracy or a specific number of iterations is reached. By leveraging this FL architecture, multiple clients can collaboratively train an SVM model while keeping their local data private. The central server facilitates the aggregation of local model parameters, enabling knowledge sharing without direct data exchange. This approach promotes privacy, data security, and distributed learning in scenarios where data cannot be centrally stored or shared due to privacy concerns or legal restrictions.

weights and bias. The equation shows that the global weights wg are obtained by averaging the updated weights $w_{l,j}^{(T)}$ of each local SVM model using the $Ud_{SVM}()$ function. Similarly, the global bias b_g is obtained by averaging the biases $b_j^{(T)}$ of each local SVM model.

Algorithm 1 The Proposed Algorithm

Data: K, T, C, X_i, y_i
Result: w, b
$w_l^{(0)} \leftarrow I_w$
$b_l^{(0)} \leftarrow I_b$
for $t = 1$ *to* T **do**
  for $i = 1$ *to* K **do**
    $w_{l_i}^{(t)} \leftarrow$ Train SVM_i on SM_i using X_i and y_i with C
  $w_l^{(t)} \leftarrow \frac{1}{K}\sum_{i=1}^{K} w_{l_i}^{(t)}$
  $b_l^{(t)} \leftarrow \frac{1}{K}\sum_{i=1}^{K} b_i^{(t)}$
  for $i = 1$ *to* K **do**
    Update SVM_i using $w_l^{(t)}$ and $^{(t)}$
    $w_{l_i}^{(t+1)} \leftarrow$ Updated weights from SVM_i
  $w_l^{(t+1)} \leftarrow \frac{1}{K}\sum_{i=1}^{K} w_{l_i}^{(t+1)}$
  $^{(t+1)} \leftarrow \frac{1}{K}\sum_{i=1}^{K} b_i^{(t+1)}$
  $w_l \leftarrow w_l^{(t+1)}$
  $b^{(t+1)} \leftarrow$ Updated value for $b^{(t+1)}$
  for $i = 1$ *to* K **do**
    Intrusion Detection:
    Obtain $\hat{y}_i$ on SM_i using X_i.
    Decision:
    if $\hat{y}_i \geq 0$ **then**
      $i \leftarrow$ *No Intrusion*
    else
      $i \leftarrow$ *Intrusion Detected*
Return w_g and b_g

Algorithm 1 aims to train a global model using data from multiple smart meters in an FL setting. The algorithm takes as input the number of smart meters K, the number of iterations T, the SVM regularization parameter C, the local dataset X_i of smart meter i, and the corresponding labels y_i. It outputs the global SVM weights w_g and bias b_g. The algorithm initializes the local SVM weights ${w_l}^{(0)}$ and bias $b_l^{(0)}$ with some initial values. Then, each iteration t from 1 to T enters a loop where it iterates over each smart meter i from 1 to K. Within this loop, the algorithm trains the local SVM model SVM_i on the local dataset X_i and labels y_i using the SVM regularization parameter C. The resulting local SVM weights are stored in $w_{l_i}^{(t)}$. After training the local models, the algorithm performs server aggregation by calculating the average of the local SVM weights $w_{l_i}^{(t)}$ and the bias $b_l^{(t)}$ across all smart meters. These aggregated weights and bias are denoted as ${w_l}^{(t)}$ and $b_l^{(t)}$, respectively. Next, the algorithm enters another loop over each smart meter i, where it updates the local SVM model SVM_i using the aggregated weights ${w_l}^{(t)}$ and bias $b_l^{(t)}$. The updated weights obtained from SVM_i are stored in $w_{l_i}^{(t+1)}$.

After updating the local models, the algorithm performs server aggregation again to calculate the average of

$$(wg, b_g) = Fed_{SVM}(X, y, K, T, C) = \left(\frac{1}{K}\sum i = 1^{K} Ud_{SVM} i\left(\frac{1}{K}\sum j = 1^{K} w_{l,j}^{(T)}, \frac{1}{K}\sum j = 1^{K} b_j^{(T)}\right), \frac{1}{K}\sum_{i=1}^{K} b_i^{(T+1)}\right) \quad (1)$$

In Eq. 1, the function $Fed_{SVM}()$ represents the FLprocess with SVM. It takes the input data X and labels y, the number of smart meters K, the number of iterations T, and the regularization parameter C. The output is the tuple (w_g, b_g) representing the final global SVM

the updated local SVM weights $w_{l_i}^{(t+1)}$ and the bias $b_i^{(t+1)}$. These aggregated weights and bias are denoted as $w_l^{(t+1)}$ and $b^{(t+1)}$, respectively. The algorithm then updates the global SVM weights w_l and bias b with the aggregated values $w_l^{(t+1)}$ and $b^{(t+1)}$. This step ensures that the global model incorporates the knowledge learned from the local models. Finally, the algorithm performs intrusion detection by iterating over each smart meter i once again. It obtains the local SVM model predictions $\hat{y}_i$ on smart meter i using the local dataset X_i. Based on the predictions, a decision is made: if $\hat{y}_i \geq 0$, the smart meter i is classified as "No Intrusion," otherwise it is classified as "Intrusion Detected." The entire process is repeated for the specified number of iterations T. At the end of the iterations, the algorithm returns the final global SVM weights w_g and bias b_g.

Proposed architecture

The proposed architecture consists of two main layers: the Fog Layer and the AMI (i.e., smart meters) Layer, as shown in Fig. 5. It represents the Fog-edge architecture for effective and timely intrusion detection without compromising data privacy. These layers work collaboratively to enable intrusion detection using SVMs in an FL setting. Each layer has distinct functionalities and contributes to the overall architecture, as described below:

1. Fog Layer: The Fog Layer represents the central computing infrastructure responsible for coordinating the intrusion detection process and hosting the global SVM model. It bridges the AMI Layer and external entities, ensuring the secure and efficient execution of the FL process. The Fog Layer handles several operations:
   (a) Initialization: The algorithm begins with initializing the Fog Layer by setting up the initial global SVM weights ($w_l^{(0)}$) and bias ($b_l^{(0)}$). These initial values serve as the starting point for the FL process.
   (b) Server Aggregation: After each iteration, the Fog Layer performs server aggregation by averaging the local SVM weights and bias contributed by the smart meters in the AMI Layer. This aggregation yields the aggregated weights ($w_l^{(t)}$) and bias ($b_l^{(t)}$), which represent the collective knowledge learned from the local models.
   (c) Global Model Update: Based on the aggregated weights and bias, the Fog Layer updates the global SVM weights (w_l) and bias (b_l). These updated values reflect the evolving global model, incorporating the insights gained from the local models in the AMI Layer.
   (d) Return of Global Model: Once the specified number of iterations is completed, the algorithm returns the final global SVM weights (w_g) and bias (b_g) obtained from the Fog Layer. These parameters can be used for intrusion detection and encapsulate the information gained during the FL process.
2. AMI Layer: The AMI Layer stands in for the network of smart meters that conducts intrusion detection. Each smart meter functions autonomously, providing local data for model training without compromising privacy or safety. The smart meters' local data feeds the global SVM model, which operates in the AMI Layer and is updated in real time. The following procedures are carried out by the AMI Layer, which consists of smart meters:
   (a) Local Model Training: Each smart meter in the AMI Layer independently trains an SVM model (SVM_i) using its local dataset (X_i) and labels (y_i), guided by the global SVM weights and the SVM regularization parameter (C). This local model training captures the peculiarities of each smart meter's data while respecting data privacy.
   (b) Local Model Update: After each iteration, the AMI Layer updates the local SVM models (SVM_i) using the aggregated weights ($w_l^{(t)}$) and bias ($b_l^{(t)}$) received from the Fog Layer. This update ensures that the local models align with the evolving global model.
   (c) Intrusion Detection: Using the updated local models, each smart meter in the AMI Layer performs intrusion detection by predicting the labels ($\hat{y}_i$) for its local dataset (X_i). Based on these predictions, the smart meter classifies itself as either "No Intrusion" or "Intrusion Detected," facilitating the identification of potential security breaches.

Experimentation

This section details the experimental setup and methodology used in the study.

Experimental setup

A computer system with an Intel Core i7-7300HQ processor, 16 GB of RAM, and a 256 SSD for faster data access is used to set up the experimentation. We used the Windows 10 operating system, Python 3.6, and the Anaconda distribution for the experiment to manage Python environments and dependencies. We used a predetermined number of Fog nodes and AMIs in the

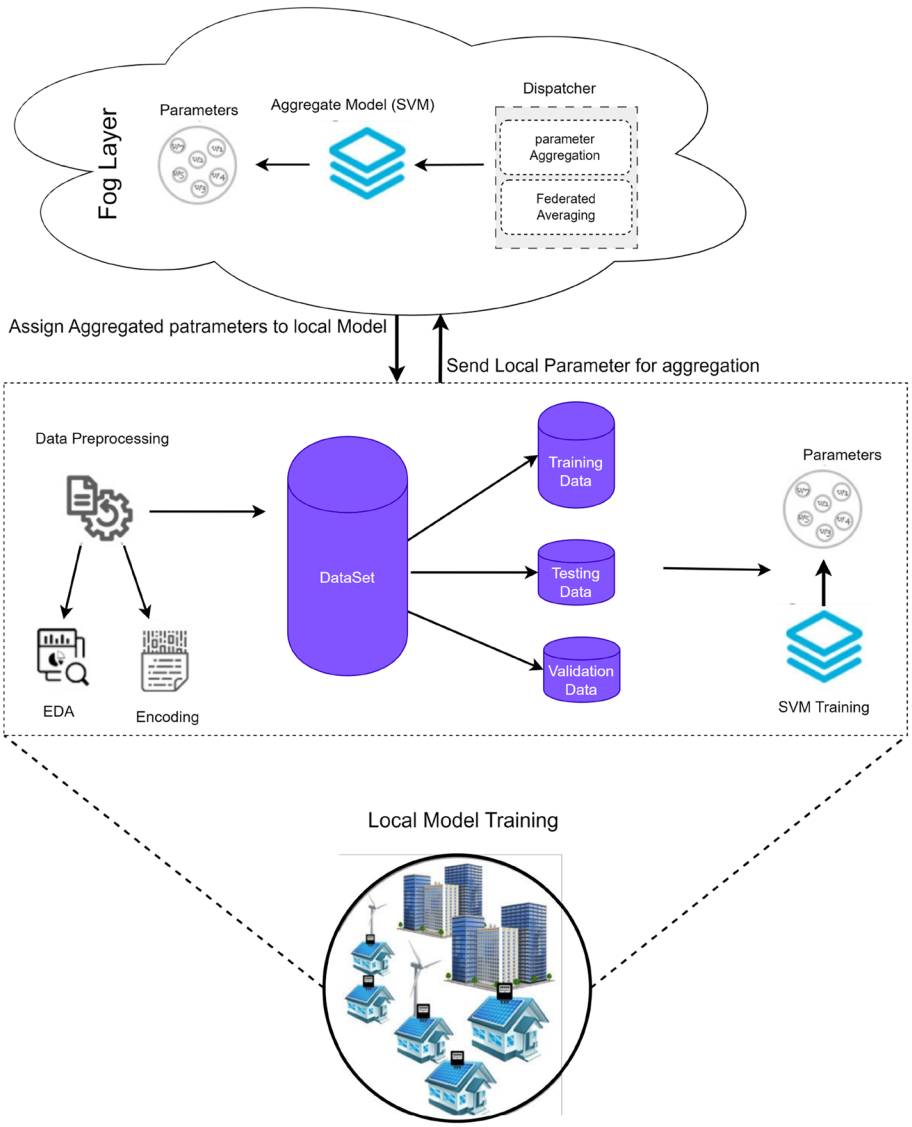

Fig. 5 Proposed Architecture

experimentation. There were 5 Fog nodes and 50 AMIs, representing a realistic scenario in an SG environment. We used Python to simulate each Fog node and AMI, where the AMIs acted as models of the regional power distribution and transmission systems used in the SG. The necessary libraries were installed to support the experiment. We used TensorFlow and Keras libraries to implement the proposed FL model. These libraries provide various resources and features for building ML models. Additionally, essential libraries were PySyft, Pandas, NumPy, and Scikit-learn.

Data preparation and model assessment may be accomplished using various ML techniques Scikit-learn provides. Pandas was used to perform effective data preprocessing and manipulation operations. While PySyft offered privacy-preserving calculations for safe cooperation in FL, NumPy provided fundamental numerical operations. We developed a customized AMI-Fog communication library to streamline the integration of Fog nodes and AMIs to facilitate data exchange and coordination between the entities using Python's native socket programming capabilities. It enabled data interchange between several simulation nodes and offered a low-level interface for network programming. The dataset was split into training, testing, and validation sets using the standard ratio of 70:10:20, and preprocessing was performed to guarantee correct formatting and handling of categorical and numerical variables. The SVM

model was initialized with 100 random values selected from a normal distribution using the preprocessed NSL-KDD dataset.

Using TensorFlow and Keras, the model was then trained to construct an SVM-based FL technique. With the help of this method, the SVM model is independently trained on each AMI in the SG. Local model updates were made on each AMI as part of the FL process, and the changes were then encrypted and securely aggregated. The global SVM model was then updated with the aggregated changes while protecting privacy. Up till convergence was reached, this iterative approach continued. A predetermined number of iterations, ranging from 10 to 50, were used to carry out the learning process. The Fog nodes and AMIs updated their local models using their respective training sets at each iteration. The privacy and data secrecy were subsequently ensured by securely aggregating the local model changes using standard cryptographic methods. To enhance the convergence and accuracy of the SVM model throughout the experiment, we changed the learning rate to 0.0001. In our investigation, it was observed that this learning rate was efficient in producing the desired outcomes. Using the testing and validation portion of the dataset, we assessed the trained model's performance once the FL process was complete. To evaluate the model's classification performance, we used a variety of assessment measures, including accuracy, precision, recall, F1 score, and sensitivity, discussed in the later section.

Methodology

This section elaborates on the proposed methodology, the flow of the model, the details of the dataset, operations on the dataset, and fog and AMI layer configuration. An abstract view of the proposed methodology flow is illustrated in Fig. 6. The AMI unit collects data in the designated region for detecting intrusions. The accumulated data is preprocessed and used to train a local SVM model. Regular updates to the local model are of utmost importance, as they ensure the preservation of its accuracy and adaptability by incorporating newly acquired data. Once the local model is trained, the parameters are sent to the global model in fog. The parameters are aggregated, and the global model uses the average for its training to enhance the accuracy and performance. The utilization of the FL approach facilitates the ability of the global model to effectively capture and encompass a wide range of insights and patterns that are relevant and applicable in various diverse contexts. As the global model progresses to be trained, the global parameters are subsequently disseminated to individual AMI units. The iterations of these models exhibit enhancements in intrusion detection in SGs. The detailed methodology and flow are explained in subsequent sections.

Data set

This study uses two renowned datasets: the NSL-KDD [80] and the CICIDS2017[1]. To handle imbalanced datasets, we adopted resampling techniques to alleviate class imbalance, such as oversampling of minority classes and undersampling of majority classes, as necessary, to ensure that the SVM models effectively learn from all classes. We employed random under-sampling, which reduced the number of majority class instances, and synthetic over-sampling using the Synthetic Minority Oversampling Technique (SMOTE) to augment the minority class. These techniques allowed us to mitigate the imbalance while preserving the integrity of the original dataset, ensuring that our intrusion detection models were trained and evaluated effectively.

The NSL-KDD data set, compared to the KDD Cup'99 data set, addresses the issue of classifier bias towards previously seen records by excluding irrelevant data from the training set. The selection of records from the KDD dataset is inversely related to their difficulty, resulting in varying classification percentages for different ML algorithms. As a result, evaluating ML methods has become more systematic and comprehensive. Moreover, the availability of record counts in both the train and test data sets allows tests to be conducted on the entire dataset, ensuring consistency in the ratings assigned to different projects.

The efficacy of the proposed model is also verified on the CICIDS2017 dataset, which is a significant resource in network security and intrusion detection. The provided dataset replicates a network configuration found in real-world scenarios, consisting of two distinct layers. The dataset comprises a significant amount of raw data, totaling 50 terabytes, recorded in PCAP (Packet Capture) files. Additionally, it includes a comprehensive collection of 84 characteristics, which are contained in CSV files. Overall, it records a substantial 2,830,743 occurrences of network traffic, facilitating thorough empirical examination.

1. Description of CICIDS2017 Dataset The dataset has been divided into 15 distinct categories, wherein 14 categories reflect various network attack methods, and one category represents benign traffic. Notably, out of the total occurrences observed, 2,273,097 can be categorized as benign traffic; however, 557,646 instances fall under the classification of anomalous. The CICIDS2017 dataset is derived from the Intru-

[1] https://www.kaggle.com/datasets/cicdataset/cicids2017/code

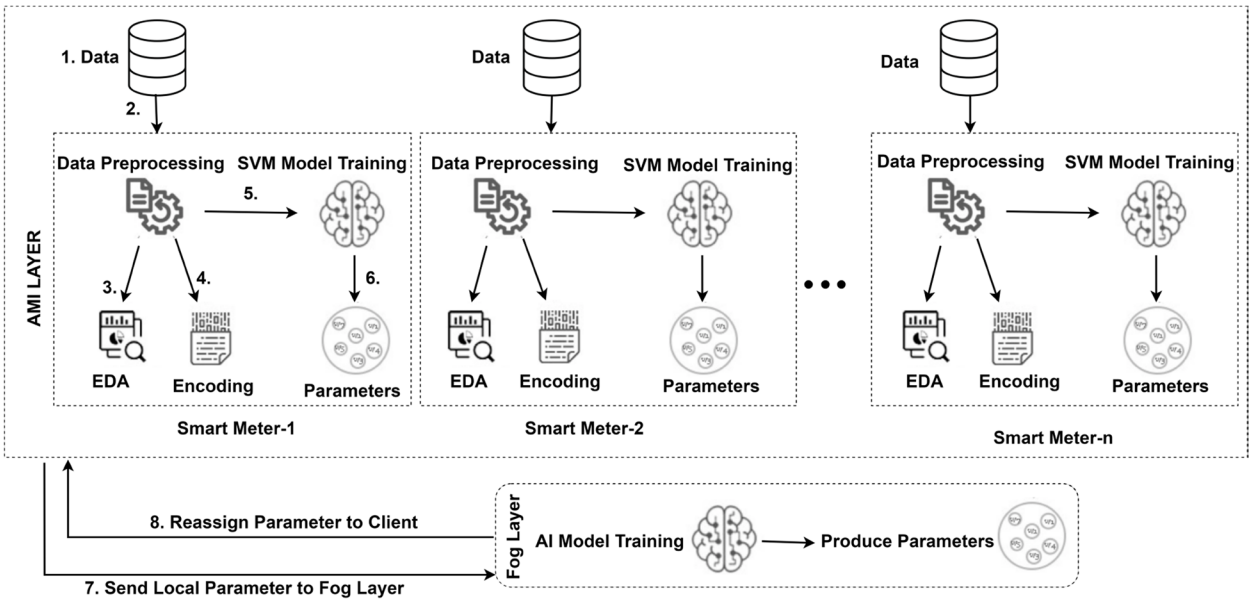

Fig. 6 Flow of the Proposed Methodology

sion Detection Evaluation Dataset. It consists of 85 attributes and covers a data-collecting period of five days, starting on July 3, 2017, and ending on July 7, 2017. The range of attacks includes Botnet attacks, Distributed Denial of Service (DDoS) attacks, Web Attacks, Denial of Service (DoS), Brute Force Infiltration, Brute Force FTP attacks, and Force SSH attacks. The CIC-IDS2017 dataset is summarised in Table 4, which includes information on different forms of traffic and their corresponding counts. The table classifies network activity, encompassing benign traffic and various forms of cyber attacks, providing a comprehensive summary of the dataset's content.

2. Description of NSL-KDD Dataset The NSL-KDD dataset is a prominent benchmark dataset for network intrusion detection. Developed as an enhanced iteration of the original KDD Cup 1999 dataset, the NSL-KDD dataset aims to rectify the limitations and biases inherent in its predecessor. This dataset encompasses network traffic data comprising both numerical and categorical attributes. The subsequent elaboration provides a comprehensive delineation of the features of the NSL-KDD dataset. For the problem at hand, encompassing 41 features, the dataset can be classified into three distinct groups: primary, content, and traffic-based. The NSL-KDD dataset is described in full in Table 5, which includes subsets such as KDDTrain that classify entries based on incursion types. Table 6 summarises the features in the NSL-KDD dataset, categorized into primary, content-based, traffic-based, and attack-type features. It is a reference for researchers and practitioners on network intrusion detection tasks. The NSL-KDD dataset's features are shown in Table 7, divided into numerical and categorical features. Examples of categorical features include Protocol Type, Service, Flag, Source IP Address, and Destination IP Address. The numerical features cover numerous parameters, including Duration, Source Bytes, Destination Bytes, and several others.

Dataset preparation

Load the dataset into an appropriate data structure, such as a pandas DataFrame, then preprocess as needed. This preprocessing may involve handling missing values, performing feature scaling, and encoding categorical variables. Studying the features and their types is essential to understand the dataset thoroughly. The dataset should be divided into multiple non-overlapping subsets, each representing a different client or participant in the FL process. These subsets should be spread across different locations or devices to ensure a fair distribution.

Preprocessing

Cleaning, standardizing, modifying, and extracting useful features from the dataset is crucial to maximize data utilization. Preprocessing the data aims to reduce dimensionality and the time required for classification, thereby enhancing the performance of classification algorithms. The outcome of data preprocessing is either

Table 4 Details of CIC-IDS2017 Dataset

Traffic Type	CIC-IDS2017	Description
BENIGN	2,273,097	Non-malicious network traffic.
Bot	1,966	Traffic from Botnets (compromised devices).
DDoS	128,027	Distributed Denial of Service attacks.
DoS GoldenEye	10,293	Denial of Service using GoldenEye tool.
DoS Hulk	231,073	Denial of Service overwhelming target.
DoS Slowhttptest	5,499	Denial of Service exploiting web server vulnerabilities.
DoS Slowloris	5,796	Denial of Service keeping many connections open.
FTP-PATATOR	7,938	Brute-force attack on FTP servers.
Heartbleed	11	Exploitation of Heartbleed vulnerability.
Infiltration	36	Unauthorized intrusion attempts.
PortScan	158,930	Scanning to discover open ports.
SSH-PATATOR	5,897	Brute-force attack on SSH servers.
WebAttack BruteForce	1,507	Web application brute-force attack.
WebAttack SQL Injection	21	SQL Injection vulnerability exploitation.
WebAttack XSS	652	Cross-Site Scripting attack.
Total	2,830,743	

Table 5 Details of NSL-KDD Dataset

Dataset Subset	Number of Records	Description
KDDTrain	125,973 entries	Training data subset
	67,343	"Normal" (Benign)
	45,927	Denial of Service (DoS)
	11,656	Unauthorized Access (R2L)
	995	Unauthorized Access to Root (U2R)
	52	Probing
Total Records	125,973	

a more concise and comprehensible representation of the desired concept or an improvement in classification accuracy or precision.

- Encoding of the datasets: The process of one-hot encoding is employed to convert categorical features in the dataset into numerical representations that ML algorithms can utilize. The following steps summarize the process of one-hot encoding for the dataset:

  – Identify Categorical Features: The dataset contains categorical features such as Protocol Type, Service, Flag, Source IP Address, and Destination IP Address.
  – Label Encoding: Categorical features must be transformed into numerical labels before applying one-hot encoding. Label encoding assigns a unique numerical value to each category within a feature.
  – One-Hot Encoding: Once label encoding is applied, one-hot encoding creates binary dummy variables for each category. Each category is represented by a separate binary feature, where a value of 1 indicates the presence of that category, and 0 indicates its absence.
  – Encoding Implementation: One-hot encoding can be implemented using libraries like pandas in Python. The commonly used function, *pd.get_dummies*(), applies one-hot encoding and generates a new data frame that includes the original numerical and encoded binary features.
  – Encoded Dataset: The resulting encoded dataset will include the original numerical features and the new binary features created through one-hot encoding. This transformed dataset can be used for further analysis or as input for ML algorithms.

Since most algorithms operate with numerical inputs, one-hot encoding enables ML models to interpret and utilize categorical data effectively. The choice to use one-hot encoding depends on the specific characteristics of the dataset and the ML algorithm employed. Other encoding methods, such as ordinal or target encoding, may be more appropriate in specific scenarios. After encoding the dataset, it is crucial to partition it randomly into non-overlapping subsets, with each subset assigned to a specific client device. This partitioning should maintain the original distribution of the dataset as closely as possible, ensuring

Table 6 Features in the NSL-KDD Dataset

Feature Category	Description
Basic Features	
Duration	The length of the connection in seconds (continuous)
Protocol Type	The protocol used in the connection (categorical: TCP, UDP, ICMP)
Service	The network service on the destination machine (categorical)
Flag	The status of the connection (categorical)
Source Bytes	The number of data bytes sent by the source (continuous)
Destination Bytes	The number of data bytes sent by the destination (continuous)
Land	Indicator of a connection from/to the same host/port (categorical: 0, 1)
Content-Based Features	
Source IP Address	The IP address of the source machine (categorical)
Destination IP Address	The IP address of the destination machine (categorical)
Source Port Number	The port number used by the source machine (continuous)
Destination Port Number	The port number used by the destination machine (continuous)
Number of Failed Logins	The count of failed login attempts (continuous)
Number of Successful Logins	The count of successful login attempts (continuous)
Number of Root Shell	The count of root shell accesses (continuous)
Number of File Creations	The count of file creation operations (continuous)
Number of Sudo	The count of sudo (superuser) commands executed (continuous)
Traffic-Based Features	
Number of Inbound Connections	The count of inbound connections to the same host/IP address (continuous)
Number of Outbound Connections	The count of outbound connections from the same host/IP address (continuous)
Number of Same Service Connections	The count of connections to the same service (continuous)
Number of Same Host Connections	The count of connections to the same host (continuous)
Number of Same Host with Same Service Connections	The count of connections to the same host and service (continuous)
Attack Types	
DoS	Denial of Service
R2L	Remote-to-Local
U2R	User-to-Root
Probing	Probing Attacks

Table 7 Table of NSL-KDD Features

Feature Type	Features name
Categorical Features	Protocol Type, Service, Flag, Source IP Address, Destination IP Address
Numerical Features	Duration Source Bytes, Destination Bytes, Num Failed Logins, Num Compromised, Num Root, Num File Creations, Num Shells, Num Access Files, Num Outbound Cmds, Count, Srv Count, Serror Rate, Srv Serror Rate, Rerror Rate, Srv Rerror Rate, Same Srv Rate, Diff Srv Rate, Srv Diff Host Rate, Dst Host Count, Dst Host Srv Count, Dst Host Same Srv Rate, Dst Host Diff Srv Rate, Dst Host Same Src Port Rate, Dst Host Srv Diff Host Rate, Dst Host Serror Rate, Dst Host Srv Serror Rate, Dst Host Rerror Rate, Dst Host Srv Rerror Rate

the representation of different classes and fair learning across client devices.

Fog and AMI layer configuration

The Fog Layer is configured to serve as the central computing infrastructure and host the global SVM model. The parameters of the SVM model, such as the regularization parameter and kernel function, are set based on prior knowledge or through cross-validation. The AMI Layer is configured to represent the distributed network of smart meters. To mitigate the risk of overfitting and underfitting in our model, we implemented several strategies while working solely with training, testing, and validation datasets. First, we incorporated L1 (Lasso) and L2 (Ridge) regularization techniques into the SVM model, introducing penalty terms in the loss function to discourage overfitting. By carefully tuning the regularization parameters, we ensured the model's

ability to generalize from the training data to unseen instances. We also ensured that our model architecture remained relatively simple, focusing on the most informative features while avoiding extraneous ones. This feature selection process not only helped the model's understanding but also mitigated overfitting risks.

Local model training

The dataset partitions are distributed to their respective client devices. On each client device, local SVM training is performed with the assigned partition of the dataset. The hyperparameters of the SVM model, such as the kernel function, regularization parameters, and other SVM parameters, are adjusted based on the requirements and constraints specific to each client device. The SVM classifier is selected as the model for FL, and its architecture, including the kernel function, regularization parameters, and other hyperparameters, is defined. The initial SVM model is distributed to all participating clients, serving as the starting point for training. Each client independently trains the SVM model on its local data using an optimization algorithm such as stochastic gradient descent, updating its local model weights based on the performance of the data. To protect data privacy, we applied differential privacy before forwarding the parameters to the global model. The sensitivity (S_n) of the model's updates is carried out using the L2 norm, and a privacy budget (P_b) of 0.1 is allocated to strike a balance between privacy and utility. We introduced Laplace noise, generated by sampling from a Laplace distribution with a scale (S_c) calculated as $S_c = \frac{S_n}{0.1}$. This scale ensured that the noise provided adequate privacy while preserving model utility. The Laplace noise is added to model parameter updates before aggregation. In managing the privacy budget, P_b is evenly allocated across multiple rounds and monitored for cumulative expenditure. If privacy usage approaches P_b, data contributions are capped to maintain privacy guarantees.

SVM components and values

SVM model training involves various components. The key components of the SVM model are as follows:

- Weight and Bias: The initial values for both the weight vector and bias were randomly set to minimal values (as low as zero) at the initialization of the FL process. This random initialization allows the SVM model to start with a neutral point, preventing any preconceived bias in the learning process.
- Kernel Function: The kernel function is a crucial component of SVM that maps the input data into a higher-dimensional feature space, enabling the non-linear separation of data points. Commonly used kernel functions include the Linear Kernel, Polynomial Kernel, and Radial Basis Function (RBF). We apply the Radial Basis Function (RBF) kernel, which is known to perform well in intrusion detection. The RBF kernel is characterized by a gamma value of 0.1.Kernel.
- Regularization Parameters: SVM incorporates regularization to control the trade-off between achieving a small margin and minimizing training errors. The primary regularization parameters are C and Gamma (for RBF kernel). Parameter C controls the penalty for misclassified training samples and the parameter Gamma defines the influence of each training sample. It also controls the trade-off between achieving a smaller margin (e.g., 'C' = 1) and minimizing training errors (e.g., 'C' = 0.1). In the context of the NSL-KDD and CICIDS2017 datasets, careful tuning of 'C' is essential to adapt the model to the specific characteristics of the data.
- Hyperparameters: In addition to the kernel function and regularization parameters, SVM models have other hyperparameters that need to be tuned for optimal performance. These hyperparameters include the kernel coefficient (for polynomial kernel), degree (for polynomial kernel), cache size, convergence tolerance, and class weights. For instance, when applying the RBF kernel, the kernel coefficient may be set to 3.
- Optimization Algorithms: SVM models are typically trained using optimization algorithms such as Sequential Minimal Optimization (SMO) or the LIBSVM library. These algorithms aim to find the optimal hyperplane that maximizes the margin between different classes while minimizing classification errors.

In our FL-based IDS, data examples consist of observations and measurements from the NSL-KDD and CICIDS2017 datasets. These data examples include information from network traffic, connection records, and related attributes. For instance, a data example may comprise details such as Protocol Type, Service, and Flag and numerical attributes like bytes transmitted and packets exchanged. Understanding the interplay between specific data features, labels, and the SVM model with actual parameter values is essential for comprehending the inner workings of our IDS, mainly when applied to the NSL-KDD and CICIDS2017 datasets. Collectively, these

components contribute to robust and privacy-preserving intrusion detection in network security environments.

The choice of the kernel function, regularization parameters, and other hyperparameters depends on the nature of the data, the problem's complexity, and the application's specific requirements. Techniques like grid search or random search can be employed for hyperparameter tuning. It is also recommended to preprocess the data, handle missing values, perform feature scaling, and apply techniques like cross-validation to ensure robust and accurate SVM model training. A secure and efficient communication protocol or framework should be established to enable communication between the central server and the clients during the FL process, ensuring data privacy and security.

Model aggregation

The clients send their updated local model weights (not the raw data) to the central server. The central server aggregates the received model weights from all clients using a defined aggregation method, such as weighted averaging, to create a new global model. The convergence of the proposed model was assessed by monitoring the stability of global model parameters over successive rounds of communication with client devices. When these parameters exhibited minimal changes, indicating that additional training rounds were unlikely to yield substantial improvements, the model was considered to have converged.

1. Federated Averaging: It is a widely used aggregation model in FL that combines locally trained models from multiple client devices or edge nodes. It enables collaborative model training while preserving data privacy. The main idea behind Federated Averaging is to leverage the local training and data from multiple client devices while maintaining data privacy and confidentiality. Instead of directly sharing raw data or gradients, client devices only send their model updates to the central server. The central server aggregates these updates through weighted averaging, improving the global model. Federated Averaging works collaboratively between a central server and multiple client devices in an FL setting. The following steps outline its operation:

    - Initialization: The central server initializes a global model with random parameters or pre-trained weights. This global model represents the initial shared model that will be collaboratively trained.
    - Local Model Training: The central server distributes the global model to participating client devices (edge nodes). Each client device trains the global model using its local dataset without sharing the raw data or gradients with the central server. The local training process typically involves multiple iterations, updating the local model's parameters using local data.
    - Local Model Updates: After training, each client device obtains updated model parameters specific to its local dataset. These updated parameters represent the local knowledge and improvements gained through training on the device's specific data.
    - Model Aggregation: The client devices send their locally trained model parameters (weights) back to the central server. The central server collects the model updates from the client devices.
    - Weighted Averaging: The central server performs weighted averaging of the received model parameters. The weights for averaging are determined based on factors such as the number of samples or computational resources of each client.
    - Global Model Update: The averaged model parameters obtained from the weighted averaging become the updated global model. The central server replaces the previous global model with the updated one.
    - Iteration: The updated global model is distributed back to the client devices for the next round of local training. Local model training, updates, aggregation, and global model updates are repeated iteratively for multiple rounds.

By repeating this process, the global model progressively incorporates knowledge from the client devices' local data while preserving data privacy. Each client device contributes local insights to the shared model without sharing sensitive data or individual gradients. This collaborative learning approach helps improve the global model's performance over time, benefiting from the collective knowledge of the participating client devices. Federated Averaging allows for decentralized training, leveraging the computational resources and data diversity across client devices, making it particularly suitable for scenarios where data privacy and locality are essential considerations.

Evaluation, testing, and validation

Evaluation, testing, and validation of the final federated model are crucial to assess its performance and determine its effectiveness in identifying intrusion. As mentioned earlier, a separate evaluation dataset is used to evaluate the model as test data for both datasets. The performance is measured using different metrics, such as

accuracy, precision, recall, or F1 score (discussed next). Once trained, our model can be deployed and used to detect intrusions on new data without further labeling. This is because the model has learned to identify the data patterns associated with intrusions.

Result analysis

The simulation results are analyzed to gain insights into the performance of the proposed architecture in intrusion detection using the NSL-KDD and CICIDS2017 datasets. The obtained performance metrics are compared to benchmark results or existing intrusion detection techniques to evaluate the effectiveness and efficiency of the proposed approach.

Accuracy

Accuracy in our experimentation refers to the model's ability to correctly classify or predict the outcome of the provided data samples (i.e., intrusions). It is calculated by dividing the number of correctly identified intrusions by the total number of intrusions made by the model. It reflects the model's performance in identifying the target variable or class labels, such as an intrusion. Figure 7 illustrates the accuracy of test and validation for NSL-KDD datasets. As the number of iterations increases, the simulation results demonstrate a clear and consistent trend of increasing accuracy for both the test and validation sets. This empirical observation suggests that enabling models to endure more extended training periods improves their predictive performance significantly. The accuracy of the test set remained 97% to 98% after 50 iterations, with an average of 98.20%, showing a minor but noticeable improvement. It was 97% to 98% after 50 iterations, with an average of 97.60% for the validation set. This progress is due to the model's capacity to learn from local data and adapt its settings to the unique features of the dataset.

In addition to the above, we followed the same FL process spanned 50 iterations for the CICIDS2017 dataset, allowing for a comprehensive examination of the accuracy trends over time. Figure 8 shows the test results and validation sets accuracy. The accuracy exhibited a notable increase from an initial 93% to a stable 96% on the test set with an average of 94.40%, indicating the convergence of the model. For the validation set, it is 93% to a stable 94% on the test set with an average of 94.40%. It shows the persistence performance of the proposed FL model. The stability observed in the accuracy trends for both datasets after a certain number of iterations suggests that the FL model, coupled with SVM, effectively converges to reliable and competitive performance levels. The ability of the model to adapt and learn from decentralized data sources is evident in the consistent accuracy improvement observed throughout the iterations. The test and validation percentages for accuracy on both datasets were comparable, with slight variations. For the CICIDS2017 dataset, the test accuracy aligned closely with the validation accuracy, showing a consistent performance of the model on unseen data. In contrast, the NSL-KDD dataset exhibited a slightly wider gap between test and validation percentages, indicating potential variations in the generalization of the model. Further investigation into the specific characteristics of the datasets and

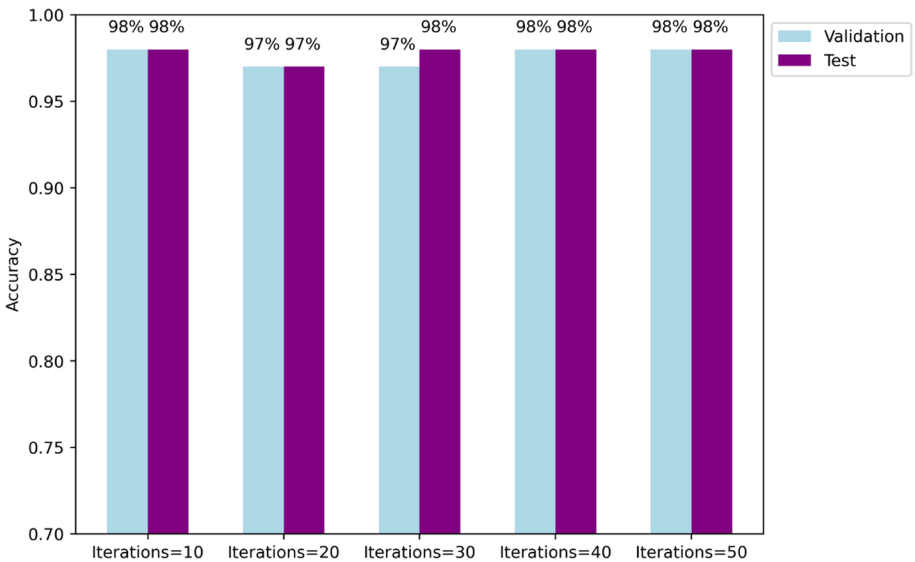

Fig. 7 Accuracy of Test and Validation sets for NSL-KDD dataset

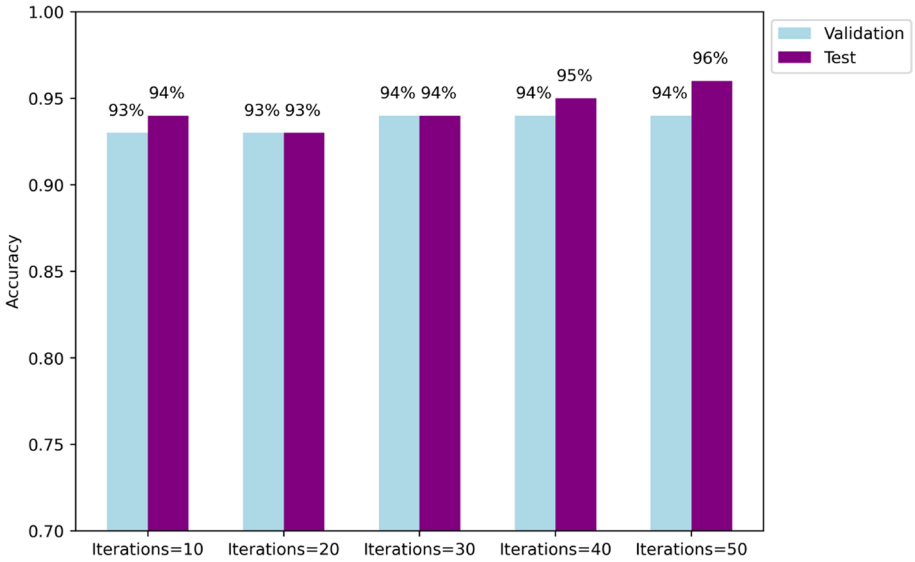

Fig. 8 Accuracy of Test and Validation sets for CICIDS2017 dataset

the model's behavior could provide insights into these differences.

Using the synergistic strengths of individual models, FL can generate substantial performance advances, as evidenced by this substantial disparity. These results demonstrate the potential of FL in real-world contexts where data privacy and distributed data sources are concerns. FL strikes the optimal equilibrium between preserving privacy and enhancing performance by allowing models to be locally trained on sensitive data and sharing only model updates.

Precision

Precision is an essential evaluation metric that verifies a model's positive predictions. It indicates the proportion of positively predicted instances (AKA true positives) correctly identified as positive among all positive instances (AKA true positives and false positives). It represents the model's ability to prevent false positives and is crucial in situations where false positives have more severe consequences, such as intrusion detection. A high level of precision indicates a small number of false positives. Figure 9 presents the test and validation sets' precision at various iterations. Analyzing the precision values, it is evident that as the number of iterations increases, both the test and validation sets exhibit an upward trend. The increase in precision indicates that the models are becoming more accurate at identifying positive instances within the dataset. For instance, the precision of the test set is 98% after 10 iterations, and the precision of the validation set is also 98%. It remained persistent throughout execution, signifying the capacity to generate precise positive predictions.

For the CICIDS2017 dataset, precision exhibited a notable increase from an initial 94% to a stable 94%, signifying the convergence of the model. The overall average precision for the CICIDS2017 dataset across all iterations is 94.4%, as shown in Fig. 10. The observed stability in precision trends for both datasets after a certain number of iterations suggests that the FL model, coupled with SVM, effectively converges to reliable and competitive performance levels. The model's ability to adapt and learn from decentralized data sources is evident in the consistent precision improvement observed over the iterations. Analyzing the differences between test and validation percentages for precision reveals nuanced insights. In the case of the CICIDS2017 dataset, the test and validation precision percentages are closely aligned, indicating consistent model performance on both seen and unseen data. Conversely, the NSL-KDD dataset exhibited a slightly wider gap between test and validation precision percentages, suggesting potential variations in the model's generalization to new and unseen data. Further exploration into dataset characteristics and model behavior can provide deeper insights into these differences.

FL's collaborative nature allows for a more varied and extensive training procedure, expanding the scope of the Global Model's coverage and improving the precision with which it predicts future events. Proof of the model's learning ability is found in their steadily improving accuracy as more iterations are added to the training process. When trained, the models

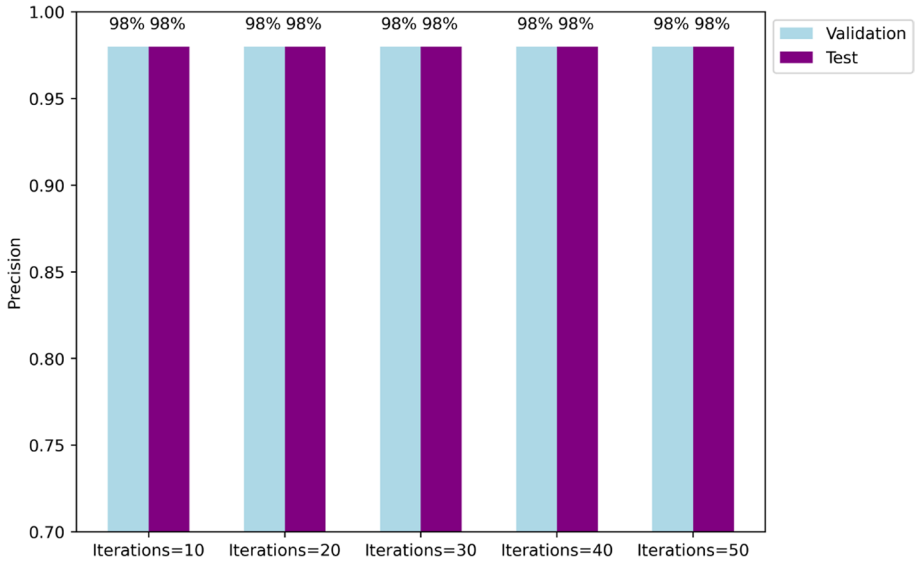

Fig. 9 Precision of Test and Validation sets for NSL-KDD dataset

improve their ability to recognize positive cases, and their parameters are tweaked. Models can improve their predictions and better capture complicated data-based interactions through this repeated learning process. The outcomes show that FL is effective in raising the models' accuracy. Through this cooperative method, the Global Model can draw upon the expertise of several regional models, improving its ability to foretell success rates. It exemplifies the promise of FL to achieve precise ML in distributed settings while maintaining privacy.

Recall

The capacity of a model to correctly identify all positive events is measured by recall (also known as sensitivity or true positive rate). The percentage of true positive examples the model correctly detects for the NSL-KDD dataset is shown in Fig. 11. The results of the memory tests conducted on the test and validation sets at different times shed light on their relative efficacy. These findings also show the importance of paying attention to each era to improve memory. The test set performed 97% recall for the 50 iterations with an average of 97%. In contrast,

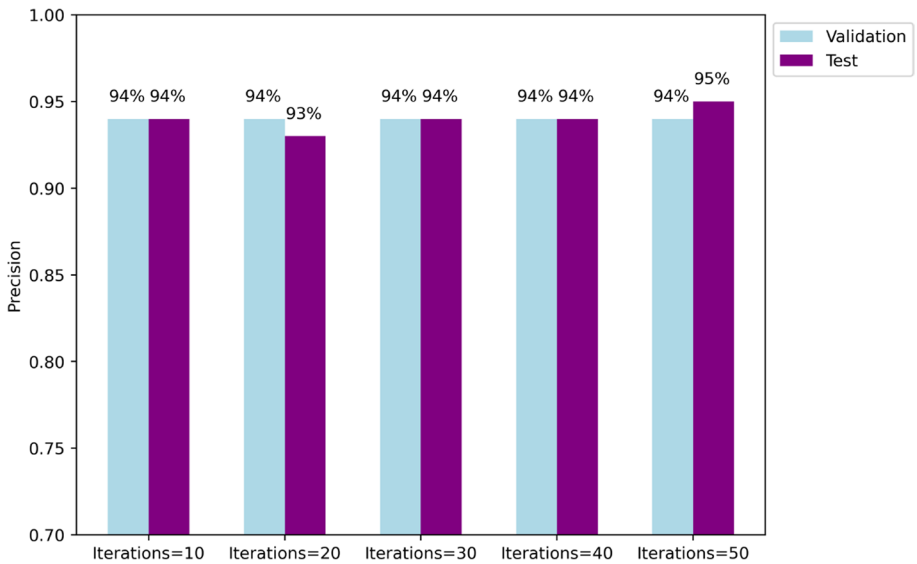

Fig. 10 Precision of Test and Validation sets for CICIDS2017 dataset

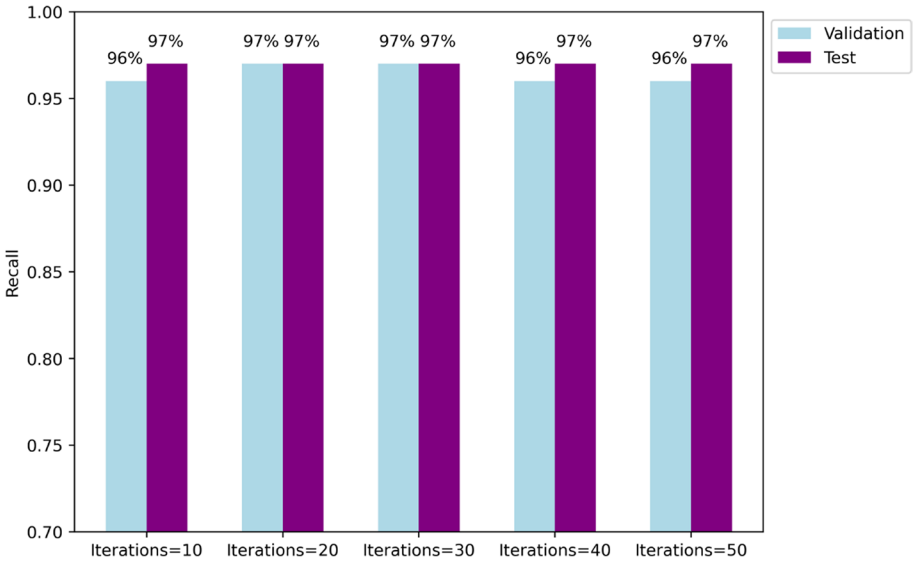

Fig. 11 Recall of Test and Validation set NSL-KDD dataset

it was 96% and 97% for the validation set, with an average of 97.20%. It indicates that the proposed models successfully identified many true positives. Similarly, recall demonstrated a stable performance for the CICIDS2017 dataset (as shown in Fig. 12), reaching 97% across all iterations. The overall average recall for the CICIDS2017 dataset is 98.6%. It was 97% to 98% recall for the 50 iterations, with an average of 97.20% for the test set. In comparison, it was 96% and 97% for the validation set, with an average of 97.66%.

The observed stability in recall trends for both datasets after a certain number of iterations indicates that the FL model, coupled with SVM, effectively converges to reliable and competitive recall levels. The model's adaptability and learning from decentralized data sources are evident in the consistent recall performance observed over the iterations. Examining the differences between test and validation recall percentages reveals interesting insights. In the case of the CICIDS2017 dataset, the test and validation recall percentages align closely,

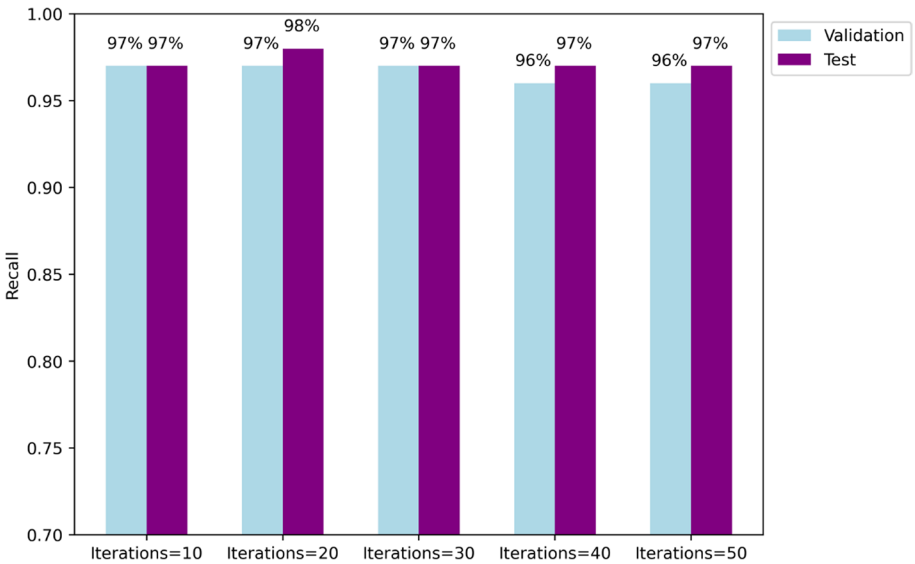

Fig. 12 Recall of Test and Validation set for CICIDS2017 dataset

indicating a consistent model performance on both seen and unseen data. Conversely, the NSL-KDD dataset exhibits a slightly wider gap between test and validation recall percentages, suggesting potential variations in the model's generalization to new and unseen data. Further exploration into dataset characteristics and model behavior can provide deeper insights into these differences. It emphasizes the necessity of leveraging the collective knowledge of distributed models to enhance intrusion detection efficiency. The recall data show how both models improve with each iteration, demonstrating how FL may successfully boost intrusion detection over time. It shows that collaborative methods are important for improving intrusion detection accuracy by achieving greater recall rates.

According to the results, the recall of both models improves as the number of iterations grows. It indicates that the models are improving at recognizing and recording success rates. When comparing recall values over time, the global model always performs better than the local model. It emphasizes the value of the global model in preventing the overlooking of potential dangers. A recall is also highlighted as an essential statistic for IDSs. The findings suggest that FL is a viable strategy for intrusion detection applications. It allows for integrating and utilizing data from several sources, resulting in more accurate and trustworthy models yet protecting individual privacy. The potential for the global model to increase the number of true positive identifications has implications for the safety of digital infrastructures. Furthermore, it enables more reliable and comprehensive identification of harmful actions.

F1 score

Both models improve in performance with each iteration, shown by a positive connection between the number of iterations and the F1 Score in Figs. 13 and 14. During this, the models might gain insights from the data and fine-tune their settings to achieve better results. With an F1 Score of 95.00% at all the iterations for the validation set, the FL Model has shown to be accurate, reliable, and stable. It means the model has successfully reduced the number of false positives and negatives while correctly detecting positive cases. The results show that the test set performs better with an F1 score of 95 to 96% with an average of 95.20%. It indicates that the model is accurate while also catching a high percentage.

The F1 Score of both models increased significantly by iteration 40. The F1 Score is used to assess the efficacy of the Local Model, which is determined to be 95%, indicating a stable model with satisfactory precision and recall. It follows a stable trend, demonstrating that the models successfully gain insights from the data and fine-tune their settings to produce the best results. The fact that the F1 Score rises with time indicates that the models get better at identifying positive events with decreasing error rates. It was successful for both the test and validation cases by striking a good balance between accuracy and recall. It shows that using different kinds of data can improve the quality of predictions made. The models' F1 scores demonstrate their trainability, suggesting they are effective for the problem.

The study guarantees the convergence of the FL model by closely monitoring performance metrics during the execution of the model. The performance of the IDS

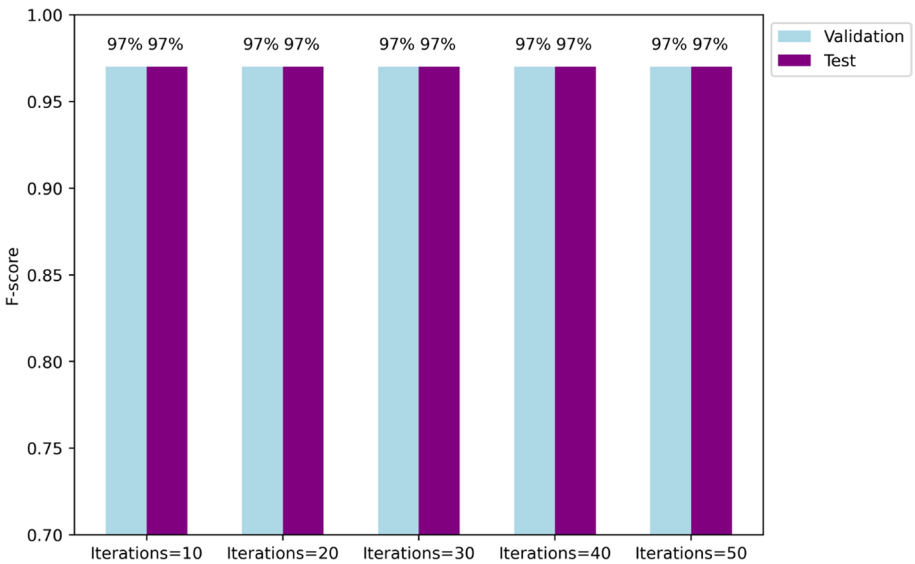

Fig. 13 F1 Score of Test and Validation sets for NSL-KDD

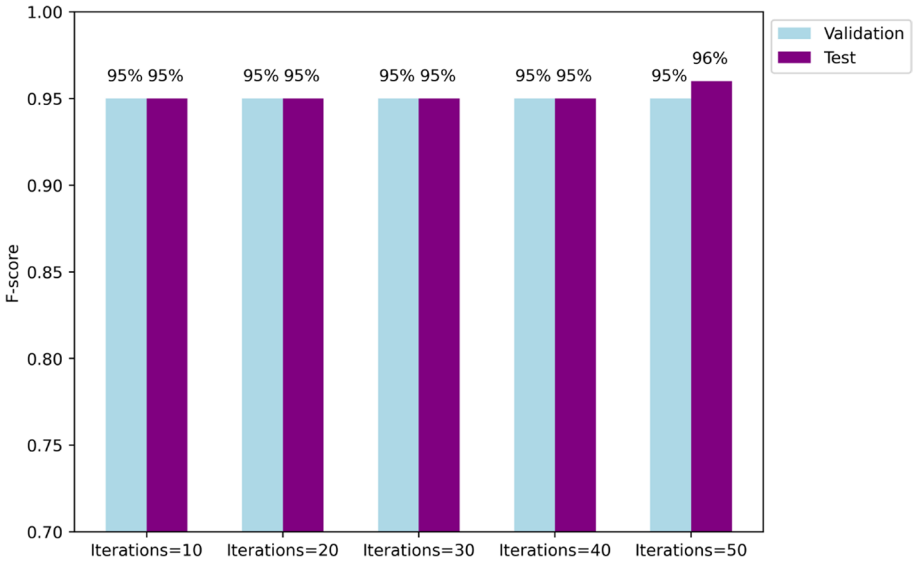

Fig. 14 F1 Score of Test and Validation set CICIDS2017 dataset

improves over time as the number of iterations increases, suggesting continual learning. However, precise convergence criteria are used to evaluate the stability of the model. An established method involves monitoring the performance parameters during the iterations. Convergence is indicated when these parameters stabilize or show diminishing improvements, even with further training iterations. Furthermore, the confirmation of convergence is strengthened by observing consistent model predictions across different devices or clients in the FL network. Performing regular model evaluation on an independent validation dataset ensures that the model has effectively learned the fundamental patterns without excessively conforming to the training data.

Confusion matrix

Observations about a classification model's efficacy can be gained from the resulting confusion matrix. Figure 15 provides crucial insights into the capabilities of the proposed IDS for SGs for the NSL-KDD dataset. The IDS captured 1,944 True Positives (i.e., 91%), demonstrating its proficiency in correctly identifying actual intrusions.

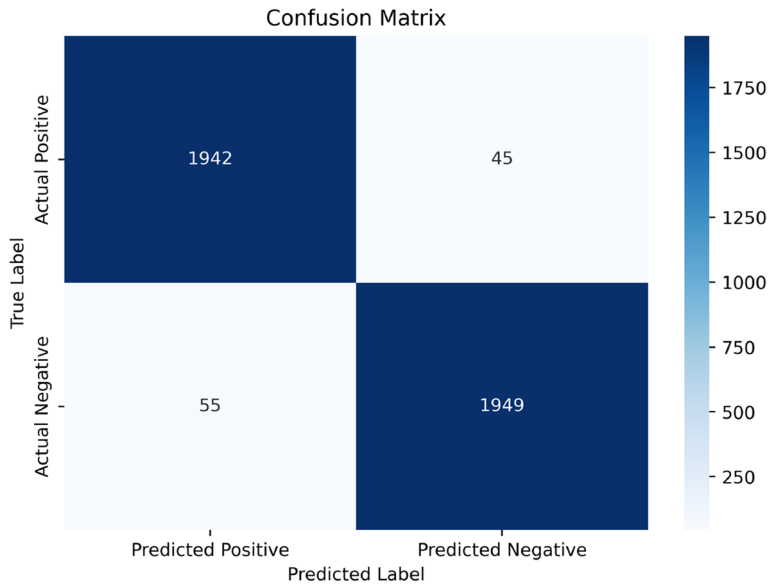

Fig. 15 Confusion Matrix for NSL-KDD dataset

Although 45 instances of False Positives were identified, indicating that normal network activity was occasionally misclassified as malicious, this number remains relatively low, indicating room for improvement. It indicates that the model sometimes incorrectly identifies normal network activity as malicious. Because SG settings are naturally complicated and changeable, good things can sometimes look like alarming trends. The False Positive Rate is 3%, meaning that three of every 1,500 occurrences when the true outcome was negative were wrongly labeled as positive. On the other hand, the IDS identified routine network traffic as non-intrusive 1,945 times, ensuring network functionality while minimizing false alarms. A count of 55 False Negatives highlights instances in which the IDS missed actual intrusions, reinforcing the need for ongoing fine-tuning to reduce false negatives further. Overall, the IDS displayed a positive indicator of its ability to differentiate between normal and malicious network activities. With a precision rate of approximately 93.26 percent and a recall rate of approximately 95.26 percent, the system excels at detecting actual intrusions and sustaining precision when issuing alerts. In addition, the comparatively low False Positive Rate (FPR) demonstrates its effectiveness in preventing superfluous alerts.

In the CICIDS2017 dataset, it is observed that 97 instances of benign network activity have been wrongly classified as malicious. It highlights potential areas that can be enhanced or improved. The misclassification observed may be attributed to the complex and ever-changing nature of SG operations. In this scenario, certain harmless activities may display patterns that resemble unauthorized access attempts. Furthermore, this dataset contains 53 instances of FN, indicating instances where the IDS could not detect real intrusions. The model has a 91% success rate in identifying 1,944 real invasions for the CICID2017 dataset, as shown in Fig. 16. It shows how well the model can identify harmful behavior. This model is effective at detecting intrusions at a 97% True Positive rate. Furthermore, the model correctly classified 1,906 instances of benign network activity. In other words, the model successfully reduced the number of unnecessary warnings while maintaining a reliable network. The model's ability to differentiate between benign and malicious traffic is shown by its high True Negative rate of 95.26 percent. There were only 97 instances in which the model incorrectly identified benign network traffic as malicious despite the model's poor False Positive rate of 3%. A minor predisposition for false alarms suggests that some disruptions and resources may be expended for no good reason. The model incorrectly labeled 53 real intrusions as harmless. Because of this, more work has to be done to perfect the detection of actual invasions. The model's low False Negative rate of 93.26% indicates that it effectively detects intrusions; however, it could be further enhanced.

Comparison with state-of-the-art

To demonstrate our model's performance and efficacy, we compared our model with a model presented in [81] for the NSL-KDD dataset. Accuracy, recall, precision, and the F1 score were some of the metrics used in the comparison (see Table 8). It was found that the proposed model had a 98% accuracy rate, which was higher than that of the FL-MGVN technique (i.e., 94%). It is to be noted that the proposed model has improved

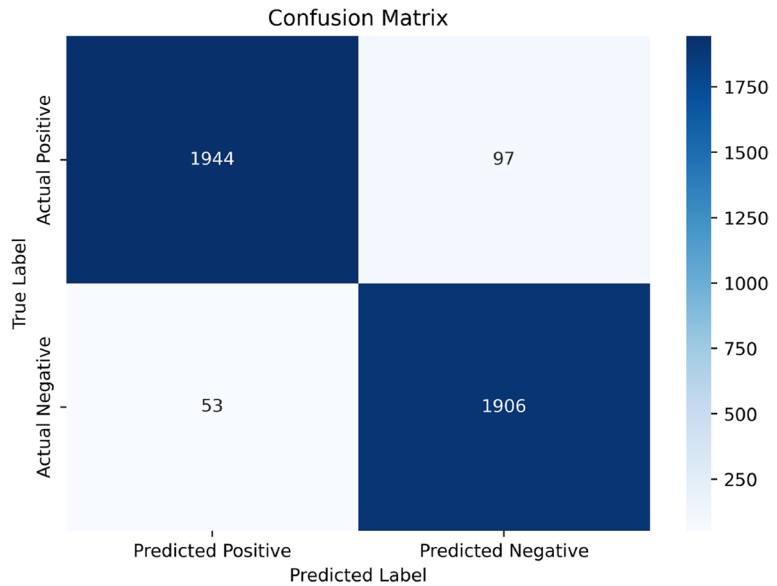

Fig. 16 Confusion Matrix for CICIDS2017 dataset

Table 8 Performance Metrics Comparison with State-of-the-art

Dataset	Metric	Paper Type		
		Base Papers	Proposed Paper	Percentage Improvement
NSL-KDD	Accuracy	94%	98%	4.17%
	Precision	89%	98%	9.63%
	F1 Score	85%	97%	13.19%
	Recall	85%	97%	13.19%
CICID2017	Accuracy	88.5%	94%	6.03%
	Precision	88.5%	94%	6.03%
	F1 Score	88.5%	95%	7.08%
	Recall	89%	96%	7.57%

classification performance across all categories. The proposed model has a 98% precision value compared to the state-of-the-art, which is only 89.0 percent. The suggested model can detect positive instances and network intrusions more accurately than the previous approaches. In addition, accurately finding positive instances is one way to measure a model's precision, which indicates the model's accuracy in labeling incursions. We also compared the two models' F1 scores. The F1 score for the proposed model was 97%, whereas the F1 score for the FL-MGVN approach was 85%. Both recall and precision are factored into the F1 score. The proposed model balanced precision and recall better, leading to more reliable performance. The base paper's recall value is 85%, whereas the proposed model scored 97% recall. Hence, the proposed method outperforms the FL-MGVN technique in every metric used in the study. The study shows that the proposed approach is better and more efficient at finding network intrusions. In contrast to the FL-MGVN, the proposed model has improved upon the accuracy, recall, precision, and F1 score benchmarks.

The performance and efficacy of the proposed model are also evaluated using the CICIDS2017 dataset. A comparative analysis is done with the state-of-the-art model presented in [82]. The model performance is assessed utilizing essential indicators, such as accuracy, recall, precision, and the F1 score, as shown in Table 8. The proposed model achieved a notable accuracy rate of 94%, surpassing the accuracy of 88.5% achieved by the state-of-the-art. The recall rate was 96%, in contrast to the state-of-the-art model, which demonstrated a comparatively lower recall rate of 89%. The significant result highlights the enhanced ability of the proposed model to reliably detect positive instances of network intrusions, surpassing the effectiveness of previous methods. Furthermore, the proposed model demonstrated a higher level of precision, with a score of 94%, compared to the state-of-the-art model's precision rate of 88.5%. Moreover, we also compared the F1 Score of the two models. The F1 score of the suggested model is 95%, whereas it achieved an F1 score of 88.5% for the state-of-the-art. As a result, the suggested model achieved a careful equilibrium between precision and recall, increasing dependability and effectiveness. Based on the findings presented, it is evident that the proposed model is superior to the state-of-the-art technique in all the metrics examined. For instance, a higher recall rate indicates a system's ability to accurately identify and mitigate potential intrusions. In practical situations, particularly those with potentially serious consequences resulting from a security breach, attaining a high recall rate assumes utmost significance. In IDSs, it is noteworthy to consider the recall rate, which can be defined as the proportion of actual intrusions that the system has correctly identified. A recall rate of 97 and 96% indicate that the system has successfully detected a high number of the total intrusions present in the system. The findings of our study represent the effectiveness of the model and also highlight its practical value in improving the security of SG networks. This study provides a clear demonstration that the proposed methodology significantly outperforms in the detection of network intrusions.

Discussion

The proposed technique is characterized by its decentralized and distributed framework, data privacy-preserving, and incorporation of a fog layer. It exhibits significant potential in addressing distinct security needs while alleviating privacy and latency problems in SGs. Additional investigation and advancement in this area may result in the creation of resilient and efficient intrusion detection models for the ever-changing SG ecosystem. One of the primary advantages of this method is its distributed and decentralized architecture. In contrast to most centralized IDSs that depend on a single point of control or data repository, this approach utilizes FL to enable multiple locations to train and enhance the intrusion detection model collectively. This strategy improves the scalability of the system. It mitigates privacy risks with centralized designs, in which a central server may have access to sensitive user data and is highly prone to a single point of failure.

The integration of fog computing optimizes the efficiency and responsiveness of SGs. Fog computing facilitates the execution of data processing and decision-making tasks at the edge of a network, resulting in decreased latency in identifying security vulnerabilities in close proximity to users. Timely intrusion detection and response in SGs is paramount since timely intervention can mitigate potentially detrimental cyberattacks. Furthermore, the FL assures that sensitive user data is kept secure on the local models/servers. The

model can undergo training without providing data to the central server, rendering it a highly suitable option for privacy-oriented SG scenarios. Therefore, it aligns with the growing significance of protecting user information in SGs. While these advantages make the proposed SVM-based FL with fog computing a compelling choice for SG intrusion detection, it is essential to acknowledge potential challenges and limitations. These may include issues related to convergence, communication overhead, and the need for careful model optimization. While fog computing offers the advantage of localized data processing and quicker decision-making at the network edge, it can introduce network delays in specific scenarios. Fog nodes may have different processing power than centralized servers, and depending on the complexity of intrusion detection algorithms and the volume of data being processed, there might be a trade-off between latency and computation time. Balancing the computational demands with low-latency response is a critical challenge in implementing fog computing for intrusion detection in SGs. Additionally, FL relies on communication between edge devices and a central server, where model updates are aggregated and shared. This communication can introduce overhead regarding network bandwidth and energy consumption, particularly in scenarios with frequent model updates. Managing this communication overhead efficiently while ensuring model convergence is a non-trivial task.

FL often involves training ML models across distributed and potentially heterogeneous data sources. Ensuring the models converge to a consistent and accurate representation of the intrusion detection task can be challenging. Variability in data distributions, quality, or device capabilities across SG nodes may hinder model convergence. Careful optimization and coordination are needed to address this limitation. As the SG network expands, the scalability of the proposed technique becomes a concern. Deploying and maintaining the necessary computational resources, including fog nodes and edge devices, can be resource-intensive. Ensuring that the system scales effectively with the growth of the SG infrastructure requires careful planning and resource management. Similarly, not all edge devices in an SG may have the computational capabilities to participate in FL effectively. Ensuring compatibility and optimizing the model for resource-constrained devices is crucial to achieving widespread adoption of the proposed technique.

While the proposed technique aims to enhance security, it also introduces new security considerations. Protecting the integrity of model updates during communication and preventing potential adversarial attacks on the FL process are vital concerns that must be addressed. Furthermore, SG data may exhibit imbalances and concept drift, where normal and attack data distribution and characteristics change over time. The technique should be able to adapt to these changes effectively and avoid becoming obsolete. While the proposed SVM-based FL with fog computing offers compelling advantages for intrusion detection in SGs, it is essential to acknowledge and address the mentioned limitations. Real-world test bed experimentation is essential for validating the effectiveness of the model in practical SG environments despite the valuable insights provided by our simulations. In addition, the effectiveness of our IDS could be improved by investigating alternative ML and deep learning models. In addition, overcoming these challenges will require a combination of algorithmic improvements, efficient communication protocols, resource management strategies, and ongoing research to ensure the technique's effectiveness and scalability in real-world SG environments. These factors emphasize the need for further investigation and empirical confirmation to strengthen the resilience of our suggested methodology.

Conclusion

Low detection accuracy, high false alarm rates, and limited labeled data availability challenge modern intrusion detection strategies. Given the escalating sophistication of cyber threats, robust IDSs play a pivotal role in safeguarding networks and protecting sensitive information. This research extends its focus to the realm of SGs, where the interconnection of devices and data communication introduces substantial security vulnerabilities. Leveraging FL and fog computing, our proposed framework offers a decentralized approach for identifying intrusions within SG environments. Our approach surpasses existing state-of-the-art methods through extensive evaluations, excelling in accuracy, recall, precision, F1 score, and specificity for detecting and classifying network intrusions. It exhibits adaptability to evolving attack patterns, scalability across diverse network environments, and an effective balance between precision and recall. In our future work, we are committed to exploring diverse network environments, evolving attack scenarios, and varied data sources while enhancing real-time capabilities and seamless integration with established security frameworks. It marks a critical step toward fortifying the security of SGs in the face of evolving cyber threats. In addition to this, we aim to integrate blockchain in the proposed solution for enhanced scalability and security. We aim to extend it to critical infrastructure and the Internet of Things (IoT) to secure diverse and interconnected systems.

Acknowledgements
The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, for funding this research work through project number 223202.

Institutional review board statement
Not applicable.

Authors' contributions
Conceptualization, Noshina Tariq and Mamoona Humayun; Data curation, Amjad Alsirhani and Faeiz Alserhani; Formal analysis, Noshina Tariq and Faeiz Alserhani; Funding acquisition, Amjad Alsirhani; Investigation, Momina Shaheen and Amjad Alsirhani; Methodology, Amjad Alsirhani; Project administration, Amjad Alsirhani; Resources, Faeiz Alserhani; Supervision, Amjad Alsirhani, Mamoona Humayun, Momina Shaheen and Faeiz Alserhani; Validation, Momina Shaheen; Writing – original draft, Noshina Tariq; Writing – review editing, Mamoona Humayun.

Funding
This work is funded by Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia.

Availability of data and materials
The data can be requested from first author using her email ID noshina.tariq@mail.au.edu.pk

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 3 December 2023 Accepted: 6 February 2024
Published online: 14 February 2024

References

1. Sun CC, Cardenas DJS, Hahn A, Liu CC (2020) Intrusion detection for cybersecurity of smart meters. IEEE Trans Smart Grid. 12(1):612–622
2. Radoglou-Grammatikis PI, Sarigiannidis PG (2019) Securing the smart grid: a comprehensive compilation of intrusion detection and prevention systems. IEEE Access. 7:46595–46620
3. Ghorbanian M, Dolatabadi SH, Masjedi M, Siano P (2019) Communication in smart grids: a comprehensive review on the existing and future communication and information infrastructures. IEEE Syst J. 13(4):4001–4014
4. Alahakoon D, Yu X (2015) Smart electricity meter data intelligence for future energy systems: a survey. IEEE Trans Ind Inform. 12(1):425–436
5. Das H, Saikia L. GSM enabled smart energy meter and automation of home appliances. In: 2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE). IEEE; 2015. p. 1–5
6. Dragičević T, Siano P, Prabaharan SS (2019) Future generation 5G wireless networks for smart grid: a comprehensive review. Energies. 12(11):2140
7. Tariq N, Asim M, Al-Obeidat F, Zubair Farooqi M, Baker T, Hammoudeh M et al (2019) The security of big data in fog-enabled IoT applications including blockchain: a survey. Sensors. 19(8):1788
8. Pedramnia K, Rahmani M, Survey of DoS Attacks on LTE infrastructure used in AMI System and Countermeasures. In: 2018 Smart Grid Conference (SGC). IEEE; 2018. p. 1–6
9. Algin R, Tan HO, Akkaya K (2017) Mitigating selective jamming attacks in smart meter data collection using moving target defense. In: Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '17. p. 1–8. https://doi.org/10.1145/3132114.3132127
10. Tufail S, Batool S, Sarwat AI. False data injection impact analysis in ai-based smart grid. In: SoutheastCon 2021. IEEE; 2021. p. 01–07
11. Chaudhry J, Qidwai U, Miraz MH. Securing big data from eavesdropping attacks in scada/ics network data streams through impulsive statistical fingerprinting. In: Emerging Technologies in Computing: Second International Conference, iCETiC 2019, London, UK, August 19–20, 2019, Proceedings 2. Springer; 2019. p. 77–89
12. Zhao J, Wang J, Yin L. Detection and control against replay attacks in smart grid. In: 2016 12th International Conference on Computational Intelligence and Security (CIS). IEEE; 2016. p. 624–627
13. Liu S, Liu XP, El Saddik A. Denial-of-Service (dos) attacks on load frequency control in smart grids. In (2013) IEEE PES Innovative Smart Grid Technologies Conference (ISGT). IEEE 2013:1–6
14. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon. 4(11):e00938. https://doi.org/10.1016/j.heliyon.2018.e00938
15. Abiodun OI, Alawida M, Omolara AE, Alabdulatif A (2022) Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: a survey. J King Saud Univ - Comput Info Sci. 34(10):10217–10245. Part B.
16. Li Y, Xue W, Wu T, Wang H, Zhou B, Aziz S et al (2021) Intrusion detection of cyber physical energy system based on multivariate ensemble classification. Energy. 218:119505
17. Khalil MI, Humayun M, Jhanjhi N, Talib M, Tabbakh TA. Multi-class segmentation of organ at risk from abdominal ct images: A deep learning approach. In: Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2021. Springer; 2021. p. 425–434
18. Hanif M, Ashraf H, Jalil Z, Jhanjhi NZ, Humayun M, Saeed S et al (2022) AI-based wormhole attack detection techniques in wireless sensor networks. Electronics. 11(15):2324
19. Moqurrab SA, Anjum A, Tariq N, Srivastava G (2023) Instant_anonymity: A lightweight semantic privacy guarantee for 5G-Enabled IIoT. IEEE Trans Ind Inform. 19(1):951–959. https://doi.org/10.1109/TII.2022.3179536
20. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR; 2017. p. 1273–1282
21. Humayun M (2021) Industrial revolution 5.0 and the role of cutting edge technologies. Int J Adv Comp Sci Appl. 12(12). https://doi.org/10.14569/IJACSA.2021.0121276
22. Wang M, Yang Z, Jiang F, Lin L, Gao M (2022) Review on offloading of vehicle edge computing. J Artif Intell Tech. https://doi.org/10.37965/jait.2022.0120
23. Namasudra S, Roy P, Balusamy B, Vijayakumar P. Data accessing based on the popularity value for cloud computing. In: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE; 2017. p. 1–6
24. Abiodun OI, Abiodun EO, Alawida M, Alkhawaldeh RS, Arshad H (2021) A review on the security of the internet of things: Challenges and solutions. Wirel Pers Commun. 119:2603–2637
25. Mohassel RR, Fung AS, Mohammadi F, Raahemifar KA, survey on advanced metering infrastructure and its application in smart grids. In: 2014 IEEE 27th Canadian conference on electrical and computer engineering (CCECE). IEEE; 2014. p. 1–8
26. Gold R, Waters C, York D (2020) Leveraging advanced metering infrastructure to save energy. American Council for an Energy-Efficient Economy (ACEEE), Washington
27. Abou El Houda Z, Hafid A, Khoukhi L. Blockchain meets AMI: Towards secure advanced metering infrastructures. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE; 2020. p. 1–6
28. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F (2021) Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol. 32(1):e4150
29. Khraisat A, Gondal I, Vamplew P, Kamruzzaman J (2019) Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity. 2(1):1–22
30. Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Comput Surv(CSUR). 2009;41(3):1–58
31. Deepa N, Pham QV, Nguyen DC, Bhattacharya S, Prabadevi B, Gadekallu TR, et al (2022) A survey on blockchain for big data: approaches, opportunities, and future directions. Future Gener Comput Syst. 131:209–226

32. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: Challenges, methods, and future directions. IEEE Signal Proc Mag. 37(3):50–60
33. Alazab M, RM SP, Parimala M, Maddikunta PKR, Gadekallu TR, Pham QV. Federated learning for cybersecurity: concepts, challenges, and future directions. IEEE Trans Ind Inform. 2021;18(5):3501–9
34. Bansal M, Goyal A, Choudhary A (2022) A comparative analysis of K-nearest neighbour, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning. Decis Anal J. 3:100071
35. Moqurrab SA, Tariq N, Anjum A, Asheralieva A, Malik SU, Malik H et al (2022) A deep learning-based privacy-preserving model for smart healthcare in Internet of medical things using fog computing. Wirel Pers Commun. 126(3):2379–2401
36. Tanveer M, Rajani T, Rastogi R. et al (2022) Comprehensive review on twin support vector machines. Ann Oper Res. https://doi.org/10.1007/s10479-022-04575-w
37. Hazra A, Rana P, Adhikari M, Amgoth T (2023) Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. Comput Sci Rev. 48:100549
38. Singh AK, Kumar J (2023) A secure and privacy-preserving data aggregation and classification model for smart grid. Multimed Tools Appl. 82:22997–23015. https://doi.org/10.1007/s11042-023-14599-4
39. Tariq N, Khan FA, Asim M (2021) Security Challenges and Requirements for Smart Internet of Things Applications: A Comprehensive Analysis. Procedia Comput Sci. 191:425–430
40. Gaur PS, Rastogi D. Analysis of the Integration of 5G with Artificial Intelligence. In: 2022 International Conference on Futuristic Technologies (INCOFT). IEEE; 2022. p. 1–6
41. Ghosal A, Conti M (2019) Key management systems for smart grid advanced metering infrastructure: A survey. IEEE Commun Surv Tutor. 21(3):2831–2848
42. Nawaz A, Hafeez G, Khan I, Jan KU, Li H, Khan SA et al (2020) An intelligent integrated approach for efficient demand side management with forecaster and advanced metering infrastructure frameworks in smart grid. IEEE Access. 8:132551–132581
43. Yao R, Wang N, Liu Z, Chen P, Sheng X (2021) Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion CNN-LSTM-based approach. Sensors. 21(2):626
44. Omolara AE, Alabdulatif A, Abiodun OI, Alawida M, Alabdulatif A, Arshad H et al (2022) The internet of things security: A survey encompassing unexplored areas and new insights. Comput Secur. 112:102494
45. Taofeek OT, Alawida M, Alabdulatif A, Omolara AE, Abiodun OI (2022) A cognitive deception model for generating fake documents to curb data exfiltration in networks during cyber-attacks. IEEE Access. 10:41457–41476
46. Abrahamsen FE, Ai Y, Cheffena M (2021) Communication technologies for smart grid: A comprehensive survey. Sensors. 21(23):8087
47. Thakre K, Goswami B. Designing neighborhood area networks of smart grid using software defined networks. In: 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE; 2021. p. 1–6
48. Yan Y, Qian Y, Sharif H, Tipper D (2012) A survey on smart grid communication infrastructures: Motivations, requirements and challenges. IEEE Commun Surv Tutor. 15(1):5–20
49. Ancillotti E, Bruno R, Conti M (2013) The role of communication systems in smart grids: Architectures, technical solutions and research challenges. Comput Commun. 36(17–18):1665–1697
50. Abiodun OI, Jantan A, Omolara AE, Dada KV, Umar AM, Linus OU et al (2019) Comprehensive review of artificial neural network applications to pattern recognition. IEEE Access. 7:158820–158846
51. Alawida M, Omolara AE, Abiodun OI, Al-Rajab M (2022) A deeper look into cybersecurity issues in the wake of Covid-19: A survey. J King Saud Univ - Comput Inform Sci. 34(10):8176–8206. Part A
52. Hooshmand MK, Hosahalli D (2022) Network anomaly detection using deep learning techniques. CAAI Trans Intell Technol. 7(2):228–243
53. Gao X, Shan C, Hu C, Niu Z, Liu Z (2019) An adaptive ensemble machine learning model for intrusion detection. IEEE Access. 7:82512–82521
54. Kush N, Foo E, Ahmed E, Ahmed I, Clark A (2011) Gap analysis of intrusion detection in smart grids. Proceedings of the 2nd international cyber resilience conference, Edith Cowan University, Perth Western Australia. https://ro.ecu.edu.au/icr/21. This Article is posted at research online.
55. Mitchell R, Chen IR (2014) A survey of intrusion detection techniques for cyber-physical systems. ACM Comput Surv (CSUR). 46(4):1–29
56. Roy DD, Shin D. Network intrusion detection in smart grids for imbalanced attack types using machine learning models. In: 2019 International Conference on Information and Communication Technology Convergence (ICTC). IEEE; 2019. p. 576–581
57. Sahani N, Zhu R, Cho JH, Liu CC. Machine learning-based intrusion detection for smart grid computing: A survey. ACM Trans Cyber-Phys Syst. 7(2):Article No:. 11. pp 1–31. https://doi.org/10.1145/3578366
58. Deng Y, Zeng Z, Jha K, Huang D (2021) Problem-based cybersecurity lab with knowledge graph as guidance. J Artif Intell Technol. https://doi.org/10.37965/jait.2022.0066
59. Namasudra S, González-Crespo R, Kumar S (2022) Introduction to the special section on advances of machine learning in cybersecurity. Comput Electr Eng. 100:108048
60. Khan LU, Saad W, Han Z, Hossain E, Hong CS (2021) Federated learning for internet of things: Recent advances, taxonomy, and open challenges. IEEE Commun Surv Tutor. 23(3):1759–1799
61. Zhang C, Xie Y, Bai H, Yu B, Li W, Gao Y (2021) A survey on federated learning. Knowl-Based Syst. 216:106775
62. Wang H, Yue W, Wen S, Xu X, Haasis HD, Su M et al (2022) An improved bearing fault detection strategy based on artificial bee colony algorithm. CAAI Trans Intell Technol. 7(4):570–581
63. Namasudra S, Nath S, Majumder A. Profile based access control model in cloud computing environment. In: 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE). IEEE; 2014. p. 1–5
64. Nassif AB, Talib MA, Nasir Q, Afadar Y (2022) Elgendy O. A systematic literature review. Artif Intell Med, Breast cancer detection using artificial intelligence techniques, p 102276
65. Cervantes J, Garcia-Lamont F, Rodríguez-Mazahua L, Lopez A (2020) A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing. 408:189–215
66. Alsokhiry F, Annuk A, Mohamed MA, Marinho M (2023) An innovative cloud-fog-based smart grid scheme for efficient resource utilization. Sensors. 23(4):1752
67. Azeem M, Ullah A, Ashraf H, Jhanjhi N, Humayun M, Aljahdali S et al (2021) Fog-oriented secure and lightweight data aggregation in iomt. IEEE Access. 9:111072–111082
68. Tariq N, Asim M, Khan FA, Baker T, Khalid U, Derhab A (2021) A blockchain-based multi-mobile code-driven trust mechanism for detecting internal attacks in internet of things. Sensors. 21(1):23
69. Tariq N, Asim M, Maamar Z, Farooqi MZ, Faci N, Baker T (2019) A Mobile Code-driven Trust Mechanism for detecting internal attacks in sensor node-powered IoT. J Parallel Distrib Comput. 134:198–206
70. Sagar BB, et al (2023) Fog Based IoT-enabled system security for Electrical Vehicles in the Smart Grids. https://doi.org/10.21203/rs.3.rs-3408518/v1. This is a preprint; it has not been peer reviewed by a journal
71. Syed NF, Ge M, Baig Z (2023) Fog-cloud based intrusion detection system using Recurrent Neural Networks and feature selection for IoT networks. Comput Netw. 225:109662
72. Sun X, Tang Z, Du M, Deng C, Lin W, Chen J et al (2022) A Hierarchical Federated Learning-Based Intrusion Detection System for 5G Smart Grids. Electronics. 11(16):2627
73. Siniosoglou I, Radoglou-Grammatikis P, Efstathopoulos G, Fouliras P, Sarigiannidis P (2021) A unified deep learning anomaly detection and classification approach for smart grid environments. IEEE Trans Netw Serv Manag. 18(2):1137–1151
74. Thirimanne SP, Jayawardana L, Yasakethu L, Liyanaarachchi P, Hewage C (2022) Deep neural network based real-time intrusion detection system. SN Comput Sci. 3(2):145
75. Husnoo MA, Anwar A, Reda HT, Hosseinzadeh N, Islam SN, Mahmood AN, et al (2023) FedDiSC: A computation-efficient federated learning framework for power systems disturbance and cyber attack discrimination. Energy AI. 14:100271
76. Liang H, Liu D, Zeng X, Ye C (2023) An intrusion detection method for advanced metering infrastructure system based on federated learning. J Mod Power Syst Clean Energy. 11(3):927–937. https://doi.org/10.35833/MPCE.2021.000279

77. Liu Y, Pi D (2017) A novel kernel SVM algorithm with game theory for network intrusion detection. KSII Trans Internet Inf Syst. 11(8). https://doi.org/10.3837/tiis.2017.08.016
78. Hossain MD, Ochiai H, Khan L, Kadobayashi Y. Smart Meter Modbus RS-485 Intrusion Detection by Federated Learning Approach. In: 2023 15th International Conference on Computer and Automation Engineering (ICCAE). IEEE; 2023. p. 559–564
79. Aashmi R, Jaya T (2023) Intrusion detection using federated learning for computing. Comput Syst Sci Eng. 45(2):1295. https://doi.org/10.32604/csse.2023.027216
80. Choudhary S, Kesswani N (2020) Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT. Procedia Comput Sci. 167:1561–1573
81. Wu D, Deng Y, Li M (2022) FL-MGVN: federated learning for anomaly detection using mixed gaussian variational self-encoding network. Inf Process Manag. 59(2):102839
82. Truong VT, Le LB (2023) MetaCIDS: Privacy-preserving collaborative intrusion detection for metaverse based on blockchain and online Federated learning. IEEE Open J Comput Soc. 4:253–266. https://doi.org/10.1109/OJCS.2023.3312299

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.