The ISC Int'l Journal of Information Security

ISeCure

October 2025, Volume 17, Number 3 (pp. 1–12)

http://www.isecure-journal.org

Presented at the ISCISC'2025 in TEHRAN, Iran.

A Federated framework for unsupervised intrusion detection on the Modbus protocol in cyber-physical systems **

Hamid Reza Dashtabadi [1], and Siavash Ahmadi [2,*]

[1] *Information Systems and Security Lab (ISSL), Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran*

[2] *Electronics Research Institute, Sharif University of Technology, Tehran, Iran*

ARTICLE INFO.

Keywords:
Industrial network intrusion detection, Modbus/TCP attacks, Federated deep learning, Autoencoder models, Raw network traffic preprocessing

Type:

doi:

Abstract

The increasing integration of modern network infrastructure into industrial control systems elevates the need for robust cyber intrusion detection for industrial protocols. Unsupervised anomaly detection is particularly effective for this task, as it identifies novel attacks by modeling normal behaviour rather than relying on limited attack data. While techniques like autoencoders, which use reconstruction error to flag deviations, can be effective, their application is often hindered by practical challenges, such as regulatory constraints and the large volumes of data that prohibit the centralised collection required for training. Federated learning offers a solution by distributing the training process to local clients and aggregating only the resulting model parameters, thus preserving data privacy and locality. This paper proposes an anomaly-based intrusion detection framework built on federated learning. Using the CIC-Modbus2023 dataset, which comprises raw Modbus traffic from a smart grid, we systematically extract and label network flows based on attack logs. We then train and evaluate several autoencoder variants—including standard, variational, and adversarial autoencoders—within this federated setting. Our results demonstrate strong performance in detecting malicious behaviour, highlighting the framework's potential as a promising approach for mitigating threats against the Modbus protocol without centralised data access. The code is available at `https://github.com/hamid-rd/FLBased-ICS-NIDS`.

© 2025 ISC. All rights reserved.

1 Introduction

Cyber-Physical Systems (CPS), which integrate cyber (e.g., computers and networks) and physical components (e.g., sensors, actuators, and controllers) are fundamental to the Industrial Internet of Things (IIoT) and Industry 4.0. In this context, industrial CPS are formed by embedding intelligent networking

* Corresponding author.

**The ISCISC'2025 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: `dashtabadi.hreza@ee.sharif.edu`, `s.ahmadi@sharif.edu`

ISSN: 2008-2045 © 2025 ISC. All rights reserved.

and computing technologies into existing industrial control systems (ICSs), enabling critical applications such as smart grids, autonomous transportation systems, and gas pipeline management [1].

This integration, however, comes with a significant tradeoff: an expanded ICS cyber-attack surface and the introduction of new risks [2]. This problem is particularly acute for legacy ICSs, which were not designed with modern security protocols and are therefore weakened by unaddressed vulnerabilities. A notable example is the Stuxnet [3] worm, which in 2012 aimed to spread throughout the Natanz industrial network (Iran) and sabotage nuclear centrifuges by exploiting some vulnerabilities in the Windows operating system. These security weaknesses are partly due to the high priority placed on availability [1] —often at the expense of integrity and confidentiality—in Operational Technology (OT) systems.

Intrusion Detection Systems (IDSs) are a critical security measure for mitigating attacks against Cyber-Physical Systems (CPSs) by monitoring network traffic or system activities for malicious behaviour [4]. These systems are broadly categorised by their data source. Host-based systems (HIDS) analyse data from a single machine, such as audit logs and system events, whereas Network-based systems (NIDS) inspect traffic across multiple hosts. A further classification is based on the detection methodology. Signature-based IDSs are effective against known attacks by matching activity against a database of predefined malicious patterns (signatures) but are inherently unable to detect novel or zero-day attacks. In contrast, anomaly-based IDSs overcome this limitation by first establishing a baseline of normal system behaviour and then flagging any significant deviation from this model as a potential intrusion, thereby enabling the detection of previously unseen threats.

While autoencoders—a class of Deep Learning (DL) models—are highly effective for network intrusion detection [5], their reliance on vast amounts of training data presents a significant challenge. Centralising this data is often impractical given the massive volume generated by multi-node CPS, alongside the unwillingness of organisations. Federated Learning (FL), introduced by Google in 2016 [6] addresses these issues of scale and privacy. In this work, we propose a network-based intrusion detection method developed within a horizontal FL framework. The key contributions of this paper include:

- **Systematic network flow extraction, preprocessing and labelling:** This contribution involves a two-stage data preparation process. First, we implement a robust feature extraction pipeline using an enhanced version of CICFlowMeter [7] to extract flows from the raw CIC-Modbus2023 dataset [8], a simulated smart grid environment. Second, we introduce a methodology to systematically label the resulting network flows by correlating them with the dataset's attack logs based on timestamps.
- **Unsupervised Learning in a Federated Setting:** We employ an anomaly-based unsupervised learning approach for network intrusion detection in industrial CPSs. This analysis involves deploying and optimising various autoencoder architectures on a benign training set and subsequently calculating their anomaly thresholds using unseen, benign validation data. We then evaluate and compare the performance of these models within a federated framework—ideal for large-scale data—using two common algorithms: FedAvg [6] and FedProx [9]. This evaluation assumes a setting where clients retain only local benign traffic, while the central server maintains attack samples for testing purposes.

The remainder of this paper is organised as follows: Section 2 reviews the existing literature on FL-based IDSs. Section 3 presents the core components of our research methodology, including the feature extraction and labelling process, the AE variants used, and the overall FL framework. Following this, Section 4 provides a detailed explanation of our proposed method. Section 5 describes the experimental setup and evaluates the results. Finally, Section 6 concludes the paper and outlines directions for future research.

2 Related Work

Nguyen [10] introduced DIoT, a self-learning distributed system designed to detect anomalous behaviours in IoT networks. The system's architecture consists of two main components: a Security Gateway (SG) and an IoT Security Service (SS). Each SG is responsible for identifying new device types and performing anomaly detection to find compromised devices. To accomplish this, each gateway locally trains a Gated Recurrent Unit (GRU) model for a specific device type and transmits it to the SS. Subsequently, the SS component aggregates the models it receives. The authors evaluated their approach using 30 IoT devices infected with Mirai, a malware used for large-scale network attacks.

Recent related works, such as DeepFed by Li *et al.* [11] and the technique by Mahmud *et al.* [12], have primarily focused on privacy-preserving supervised FL for intrusion detection in industrial systems. Both

[1] This represents a reversal of the conventional CIA triad (Confidentiality, Integrity, Availability).

approaches use a hybrid model architecture combining Convolutional Neural Networks (CNNs) and GRUs. DeepFed, for instance, was experimentally validated on a sensor-based timeseries dataset and employed a Paillier-based cryptosystem for security.

The study in [13] proposed an ensemble method where a random forest classifier aggregated outputs from several Recurrent Neural Network (RNN) models. For their experiments, they used the ICS dataset [14], which contains Modbus protocol network traffic from an emulated system—a scenario analogous to our own. This dataset includes five traffic categories: Clean, Man-in-the-Middle (MITM), Ping DDoS Flood, Modbus Query Flood, and TCP SYN DDoS Flood.

While the authors reported that their FL approach achieved higher accuracy than a non-FL baseline, two aspects of their methodology warrant consideration. Features were extracted using CICFlowMeter [15] from entire capture files, which were then labelled based on their filenames (e.g., "attack"). This method may have mislabelled intervals of benign activity within an attack capture as malicious, potentially influencing the reported accuracy. Second, since CICFlowMeter does not support the ICMP protocol, all packets related to the Ping DDoS Flood attack were likely omitted during the flow extraction.

Unlike approaches that rely on RNNs or CNNs to detect contextual or collective anomalies in sequential sensor or packet-level data, our method employs point anomaly detection, which is more robust for network flow analysis. This focus is advantageous because attackers can disrupt or displace flow sequences, potentially evading sequential detection models. Furthermore, our framework prioritises mitigating the high communication load from large-scale local datasets in FL, rather than focusing on cryptographic privacy-preservation mechanisms often paired with sequential models.

Other studies have also focused on FL-based NIDS. For instance, Aouedi *et al.* [16] recently proposed a FL scheme that leverages a semi-supervised approach. In their framework, an autoencoder is first trained on each client using local unlabelled data. The central server then aggregates these into a global autoencoder via FL. Finally, the server constructs a supervised classifier by adding fully connected layers to the global encoder, which is then trained using publicly available labelled data. In subsequent work, [17] introduced a low-complexity supervised FL framework designed for intrusion detection and attack classification within Software-Defined Networking (SDN)-based CPS. To mitigate model complexity and enhance performance, they employed the Chi-square and Pearson correlation coefficient methods for feature selection.

The work in [18] proposed a Deep Neural Network (DNN) model for supervised anomaly detection in Distributed Energy Resources (DER), leveraging both horizontal (HFL) and vertical (VFL). The model accounts for crucial DER communication protocols such as DNP3 and Modbus. For their evaluation, the authors used the CIC-Modbus2023 dataset. The feature engineering process involved using the CICFlowmeter library to extract 72 statistical IT/OT features, followed by dimensionality reduction to 43 features using Principal Component Analysis (PCA). Their model achieved high accuracy, F1-score, and recall, with metrics approaching 90%. However, the study does not specify the methodology used for flow labelling.

A core tenet of these methods is their reliance on supervised or semi-supervised learning, which necessitates labelled datasets for training. Our method is partly inspired by the study in [19], which deployed Fed-ANDIS, an anomaly-based unsupervised NIDS using FedAvg and FedProx. In their approach, they utilised AE variants to learn a semantic representation of normal traffic samples. After the learning phase, a detection threshold was calculated based on the reconstruction error on an unseen validation set of normal data. This threshold was then used to evaluate the model's performance on a labelled test set containing attack samples. On the CIC-IDS2017 dataset [20], they achieved notable F1-scores of up to 93% for a standard AE, 62% for a VAE, and 80% for an AAE.

The existing literature presents two key opportunities for contribution. First, there is no documented, systematic pipeline for transforming the raw CIC-Modbus2023 dataset into a machine learning-ready format. Concurrently, the potential of employing unsupervised, flow-based point-anomaly techniques for industrial network security within an FL framework is a significant and underexplored area of research.

3 Autoencoder-based FL framework

This section details our method, commencing with a description of the dataset, its network architecture, and the specific Modbus protocol attack techniques it contains. Subsequently, we introduce the autoencoder variants employed in this work and briefly outline their theoretical foundations. Finally, we present the two FL approaches—FedAvg and FedProx—that orchestrate the distributed training for intrusion detection.

3.1 Dataset

The Modbus protocol [21], widely utilised in industrial systems, operates on a client/server (historically master-slave) architecture. Although initially designed for serial communication, it has been adapted for use over the Transmission Control Protocol (TCP) in a version known as Modbus/TCP. The structure of its data unit format is shown in Figure 1. Within this framework, a client, such as a SCADA Human-Machine Interface (HMI), issues queries using specific function codes (e.g., read/write for single or multiple registers). The server, in this case an Intelligent Electronic Device (IED), then processes the request and returns a corresponding response. The

MBAP Header				PDU	
Trans. ID (2B)	Proto. ID (2B)	Length (2B)	Unit ID (1B)	Func. Code (1B)	Data (nB)
Modbus TCP ADU					

Figure 1. A Modbus/TCP message, known as an Application Data Unit (ADU), consists of a Modbus Application Protocol (MBAP) Header and a Protocol Data Unit (PDU)

CIC-Modbus2023 dataset contains Modbus/TCP network traffic captured using Wireshark within a simulated electrical substation. Physical simulation centred around tap changing of a voltage transformer. This testbed environment was built with Docker containers to emulate key industrial components, including IEDs and a SCADA HMI. The dataset features a variety of simulated attacks against the Modbus protocol (application layer), which are categorised by their origin: external attacker, compromised SCADA master, or a compromised IED. Table 1 summarises the attack methods within the dataset.

3.2 Autoencoders

As illustrated in Figure 2, an **Autoencoder(AE)** [22] is an unsupervised neural network comprising two main components: an encoder and a decoder. The encoder learns a compressed, low-dimensional latent representation of the input data. Given an input vector $x \in \mathbb{R}^d$, the encoder maps it to a latent vector $z \in \mathbb{R}^k$, where the latent dimension is significantly smaller than the input dimension ($k \ll d$). The decoder then attempts to reconstruct the original input from this latent z, producing an output vector $\hat{x}$ with minimal loss error.

A **Variational Autoencoder (VAE)** [23] is a generative version of the simple AE. Instead of mapping an input directly to a single latent vector, the VAE encoder $q(z|x)$ maps it to the parameters of a probability distribution—typically a Gaussian distribution defined by a mean vector (μ) and a standard deviation vector (σ). A latent vector z is then randomly sampled from this learned distribution ($z \sim$

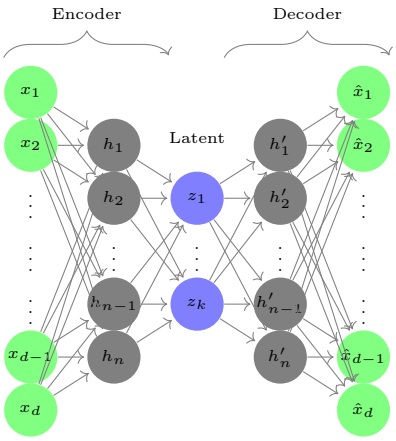

Figure 2. The architecture of a simple autoencoder (AE) consists of an encoder and a decoder block

$\mathcal{N}(\mu, \sigma^2)$). The decoder $p(x|z)$ uses this sampled vector z to reconstruct the original input. This process allows the VAE not only to reconstruct data but also to generate new data by sampling from the latent space.

The VAE is trained by minimising a loss function derived from the Evidence Lower Bound (ELBO), which consists of two key components. The first term is a reconstruction loss that, similar to a simple AE, ensures the faithful reconstruction of the original input. The second component, the Kullback-Leibler (KL) divergence, constrains the learned distribution $q(z|x)$ to approximate prior $p(z)$, creating the continuous latent space necessary for generating high-quality samples. The combined loss function is:

$$\mathcal{L}_{\text{VAE}} = \underbrace{-\mathbb{E}_{q(z|x)}[\log p(x|z)]}_{\substack{\text{Reconstruction} \\ \text{Loss}}} + \underbrace{D_{KL}(q(z|x)||p(z))}_{\substack{\text{KL} \\ \text{Divergence}}}$$

Adversarial Autoencoders (AAEs) [24] are hybrid models that combine Autoencoders (AEs) and Generative Adversarial Networks (GANs) [25]. An AAE, based on a min-max game, aims for latent space alignment, compelling its aggregated posterior distribution, $q(z)$, to match a predefined prior, $p(z)$. The networks are trained jointly via Stochastic Gradient Descent (SGD), alternating between two main phases: reconstruction and adversarial training.

(1) Reconstruction Phase: The autoencoder (encoder and decoder) is first updated to minimise the reconstruction error between an input x and its decoded output $\hat{x}$.
(2) Adversarial Training Phase: An adversarial process is used to enforce a prior distribution, $p(z)$, on the latent space. This phase consists of two steps that are performed iteratively:
First, the discriminator (D) is trained to distinguish "real" samples drawn from the prior

Table 1. Consolidated summary of Modbus/TCP attack techniques

Attack technique	Description
Reconnaissance	Probes the network by sending read queries with various function codes across a wide range of addresses. This is used to map out active devices and discover valid data registers for the next phases of a probable attack vector.
Flooding Attack	Overwhelms a target device (either an IED or the HMI/master) by sending a high-volume burst of packets. This is executed by either sending a rapid succession of queries (query flooding) or a large number of responses to a single request (response flooding).
Length Manipulation	Alters the Modbus/TCP length header field in a query or response packet to an invalid value. This attack aims to cause parsing errors, crash the receiving application, or evade detection by protocol-aware security tools.
Replay Attack	Maliciously retransmits previously captured packets. This includes resending legitimate queries to evade detection(sometimes in a last-in-first-out sequence) or replaying cached, valid responses to mask an ongoing fault or attack from the HMI.
Payload Injection	Manipulates the data portion of a Modbus packet to cause unintended behaviour. Techniques include subtly modifying query parameters (e.g., changing the number of registers to read), appending arbitrary bytes to a packet, or replacing legitimate data in a response with completely fabricated values.
Frame Stacking	Embeds multiple Modbus payloads, or Application Data Units (ADUs), within a single TCP packet. This non-standard communication can be sent in either a query or a response to test or exploit vulnerabilities in the target's protocol parser.
Brute Force	Employs systematic, repetitive attempts to compromise a device. Methods include cycling through a vast range of transaction IDs to execute a command, or sending queries with malformed payloads or requests for excessively large data ranges (e.g., 2000 bits of a coil) to trigger system errors such as "Illegal Data Address".
Delay Response	A compromised IED intentionally delays responses to force the HMI to wait, causing a denial of service. This disrupts the system, preventing the transmission of requests to other IEDs and the central agent.

$p(z)$ from "fake" samples generated by the encoder $q(z|x)$. The discriminator uses the standard binary cross-entropy objective to calculate its loss:

$$\mathcal{L}_{\text{discriminator}} = -\mathbb{E}_{z\sim p(z)}[\log D(z)] - \mathbb{E}_x[\log(1 - D(q(z|x)))]$$

Second, the generator (encoder) is trained to fool the discriminator, thereby shaping the latent space to match the prior distribution. The generator's loss is:

$$\mathcal{L}_{\text{generator}} = -\mathbb{E}_x[\log D(q(z|x))]$$

In practice, Monte Carlo estimation [26] is a computational technique used to estimate expectations by randomly sampling from the data distributions.

3.3 Federated learning

Federated Averaging (FedAvg) [6] is a foundational FL algorithm in which a central server aggregates model parameters trained locally on client data, iteratively building a global model without accessing the raw data. **FedProx** [9] extends FedAvg by adding a proximal term to the local objective function, thereby mitigating issues arising from data heterogeneity.

4 Proposed method

We propose a distributed system in which clients preprocess their raw local network traffic, train anomaly-based intrusion detection models locally, and send their optimised parameters to a server. The server subsequently aggregates these parameters and evaluates the model using a limited set of attack samples. The steps involved are 1) Global initialisation, 2) Preprocessing and labelling, 3) Local Training and 4) Aggregation and evaluation. Each step is explained in detail in the remainder of this section.

4.1 Global initialisation

The FL process commences with an initialisation phase orchestrated by the central server. Initially, a specific autoencoder architecture—either AE, VAE, or AAE—is selected. The model's parameters are initialized using the Xavier method [27], a technique pivotal for ensuring training stability by preserving the variance of activations during the forward pass and gradients during the backward pass. Concurrently, essential hyperparameters, including the learning rate (lr) and the total number of communication rounds (n_r), are defined. These parameters are then shared with the selected clients, initiating the FL process.

4.2 Preprocessing and labelling

To prepare the data for machine learning tasks, a set of 87 statistical features was extracted from raw packet capture (.PCAP). This process involves grouping packets into network flows, where a flow is defined as a sequence of related packets sharing the same source/destination IP addresses, port numbers, and protocol type (e.g., TCP).

Our packet processing pipeline adheres to the guidelines from Idressi et al. [19]. First, we repair any po-

tentially damaged capture files using the pcapfix [2] tool. Subsequently, we apply the reordercap [3] on the resulting PCAP files to sort all packets by timestamp, ensuring the temporal integrity of the data.

For network flow feature extraction, we employ an enhanced version of the CICFlowMeter tool [7]. This process initially extracts a comprehensive set of 89 features for each flow, encapsulating detailed traffic characteristics. These include statistical metrics such as flow duration, packet counts, and sizes for both forward and backward directions, and inter-arrival times. Non-statistical identifiers, namely IP addresses and port numbers, are temporarily retained alongside the statistical features, as they are essential for the subsequent labelling process. Upon completion of this phase, the entire feature set is exported to Comma-Separated Values (CSV) format.

A multi-stage preprocessing pipeline is then applied to the extracted data. First, the non-statistical identifiers are discarded. The remaining numerical features are subsequently normalised to the interval $[0, 1]$ using min-max scaling. Next, feature selection eliminates features exhibiting zero variance, yielding a refined set of 76 statistical features. Finally, the data undergoes necessary encoding and is structured into batches for input into the machine learning models.

The labeling process began by identifying candidate flows through the correlation of IP addresses and event timestamps from the logs with each flow's start and end timestamps. To perform this matching efficiently, flows were sorted by their start timestamp. This arrangement enabled the use of the `bisect_left` [4] search algorithm to filter for flows with start and end times in proximity—within a reasonable tolerance—to the start and completion of the logged attack event, with the first match being automatically selected. Once a match was found, the flow was assigned the appropriate label from the 'Attack' column in the log file. Finally, to streamline the classification, labels representing similar attack concepts were merged. To ensure the accuracy of this automated process, the assigned labels were randomly verified against the packets in the corresponding PCAP files and attack logs for all scenarios. This verification revealed that attack logs for day 21 of the 'compromised-scada' scenario (ied1b) had been missed, prompting the removal of this data. As an example, Figure 3 shows the resulting label distribution for this node in the 'compromised-scada' scenario.

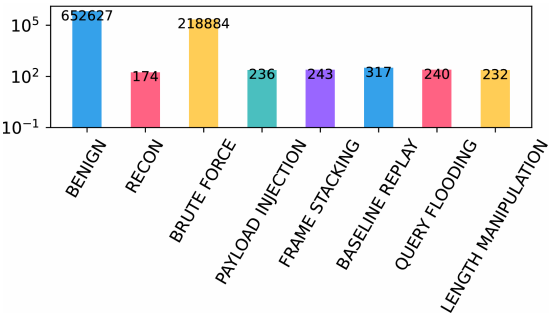

Figure 3. Comrpomised-scada scenario attack label distribution in IED node ied1b

4.3 Local Training

Each client downloads the global model parameters (θ^g, ϕ^g) to initialize its local autoencoder, parameterized by (θ, ϕ). The client then optimises its local model parameters on its preprocessed dataset D for n_e epochs. The training process minimises a compos-

Algorithm 1 Client-Side Local Training

function LocalUpdate($\theta^g, \phi^g, \Omega^g, \dots$)
Input: Global params $\theta^g, \phi^g, \Omega^g$,$lr, wd, n_e, \mathcal{D}, B, \lambda$
Output: Updated local params θ, ϕ, Ω
Init from global $\theta \leftarrow \theta^g, \phi \leftarrow \phi^g, \Omega \leftarrow \Omega^g$
for $e = 1$ **to** n_e **do**
for each batch $\boldsymbol{x} \subset \mathcal{D}$ **do**
if AE or VAE **then**
$\hat{\boldsymbol{x}} \leftarrow p_\theta(q_\phi(\boldsymbol{x}))$ // AE pass
$L_{rec} \leftarrow \frac{1}{B}\sum\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 + wd \cdot (\|\theta\|^2 + \|\phi\|^2)$
if VAE **then**
// Approximate posterior parameters
$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \Rightarrow$ mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$.
$L_{KL} = D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p(\boldsymbol{z}))$
$= \frac{1}{2}\sum\left(\boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - \mathbf{1}\right)$
(for $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$)
$L_{rec} += L_{KL}$
end if
$L_{rec} += \frac{\lambda}{2} \cdot (||\theta - \theta^g||^2 + ||\phi - \phi^g||^2)$
$\theta, \phi \leftarrow \theta - lr\nabla_\theta L_{rec}, \phi - lr\nabla_\phi L_{rec}$
else if AAE **then**
$\boldsymbol{z_p} \sim p(z)$
$\boldsymbol{z_f}, \hat{\boldsymbol{x}} \leftarrow p_\theta(q_\phi(\boldsymbol{z}|\boldsymbol{x}))$
// 1) Train discriminator:
$L_{dis} \leftarrow \frac{1}{B}\sum[\log D_\Omega(\boldsymbol{z_p}) + \log(1 - D_\Omega(\boldsymbol{z_f}))]$
$\Omega \leftarrow \Omega - lr \cdot (L_{dis} + wd \cdot \|\Omega\|^2 + \frac{\lambda}{2} \cdot \|\Omega - \Omega^g\|^2)$
// 2) Train generator:
$L_{rec} \leftarrow \frac{1}{B}\sum\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$
$L_{gen} \leftarrow \frac{1}{B}\sum(1 - \log(D_\Omega(\boldsymbol{z_f})))$
$L_{\text{gen}} \leftarrow L_{\text{rec}} + wd \cdot (\|\phi\|^2 + \|\theta\|^2) + \frac{\lambda}{2} \cdot (\|\phi - \phi^g\|^2 + \|\theta - \theta^g\|^2)$
$\theta, \phi \leftarrow \theta - lr\nabla_\theta L_{gen}, \phi - lr\nabla_\phi L_{gen}$
end if
end for
end for
return θ, ϕ, Ω (Send to server)
end function

[2] https://f00l.de/pcapfix/
[3] https://www.wireshark.org/docs/man-pages/reordercap.html
[4] https://docs.python.org/3/library/bisect.html

ite loss function, with the primary component being the **Mean Squared Error (MSE)** reconstruction loss. This loss, defined as $\mathcal{L}_{rec} = \|x - \hat{x}\|^2$, quantifies the discrepancy between an input vector x and its reconstructed output $\hat{x}$.

For VAE, the objective function includes a KL divergence term, $\mathcal{L}_{KL} = D_{KL}(q_\phi(z|x) \,\|\, p(z))$. This term acts as a regularizer, encouraging the learned posterior distribution $q_\phi(z|x)$ from the encoder to approximate a standard isotropic Gaussian prior, $p(z) \sim \mathcal{N}(0, I)$. In contrast, AAE uses an adversarial training objective. The Discriminator Ω is trained to distinguish between latent codes from the encoder and samples drawn from the prior $z_p \sim p(z)$. The encoder is simultaneously trained to produce fake $z_f \sim q_\phi(z/x)$ that fool the discriminator, thereby implicitly imposing the prior distribution on the latent space.

To mitigate overfitting,a L2 regularisation term [28], $\mathcal{L}_{reg} = wd \cdot \|w\|^2$, is applied to the local model's weights w . Finally, in the FedProx configuration, the objective is augmented with a proximal term, $\mathcal{L}_{prox} = \frac{1}{2}\lambda\|w - w^g\|^2$. This term penalises large deviations of the local model w from the global model w^g, with λ as a controlling hyperparameter. After completing the local training, each client sends its optimised parameters, w, to the server for aggregation. The client-side procedure is detailed in Algorithm Algorithm 1.

4.4 Aggregation and Evaluation

Upon receiving updates from all clients, the server updates the global model's parameters by computing their weighted average. Subsequently, a global anomaly detection threshold is established using the K-SIGMA method [29]. This process involves each client first calculating a local threshold, thr, on its respective validation dataset of unseen, normal network flows. The server then aggregates these local thresholds to establish the global threshold, as shown in Figure 4. Each client's local threshold is defined as: $thr = \mu + k \cdot \sigma$ where μ and σ are the mean and standard deviation of the MSE reconstruction losses on the validation set, and k is a tunable hyperparameter. The K-SIGMA method was chosen for its computational efficiency and widespread use, providing a clear baseline to assess our FL model's performance. Although this static approach has known limitations in dynamic environments, its simplicity allows us to directly evaluate our primary contribution.

During inference, the **Intrusion Score** (IS) for a given flow, defined as its reconstruction loss, is compared against this threshold. A flow is classified as an intrusion if $IS > thr$ and benign otherwise. The server evaluates the global model's efficiency on a held-out attack test dataset using appropriate classification metrics. Following the evaluation, the server distributes the updated global model back to the clients to commence the next round of training. This iterative process continues for a predefined number of communication rounds, n_r, or until a target performance level is met. The complete server-side procedure is detailed in Algorithm 2.

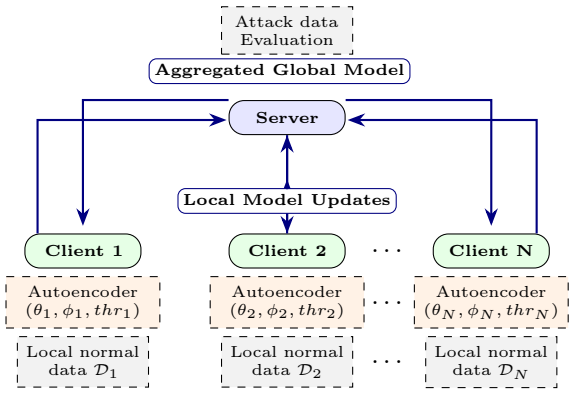

Figure 4. Client-Side Training and server-side evaluation using aggregated parameters and thresholds

Algorithm 2 Server Aggregation and Evaluation

Require: Initial model $(\boldsymbol{\theta}_g^0, \boldsymbol{\phi}_g^0, \boldsymbol{\Omega}_g^0)$; rounds n_r; hyperparameters $(\lambda, n_e, k, \dots)$.
for each round $r = 0, \dots, n_r - 1$ **do**
// On Clients:
For each client $c_i \in C$
$(\boldsymbol{\theta}_i^{r+1}, \boldsymbol{\phi}_i^{r+1}, \boldsymbol{\Omega}_i^{r+1}) \leftarrow \textsc{LocalUpdate}(\boldsymbol{\theta}_g^r, lr, \dots)$
$thr_i^{r+1} \leftarrow \mu_{\text{loss}} + k \cdot \sigma_{\text{loss}}$ from $D_{valid,i}$.
// Server-side Aggregation:
$N_{\text{train}} \leftarrow \sum_{i \in C} |D_{\text{train},i}|$
$N_{\text{valid}} \leftarrow \sum_{i \in C} |D_{\text{valid},i}|$
$\boldsymbol{w}_g^{r+1} \leftarrow \sum_{i \in C} \frac{|D_{\text{train},i}|}{N_{\text{train}}} \boldsymbol{w}_i^{r+1}, \boldsymbol{w} \in \{\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\Omega}\}$
$thr_g^{r+1} \leftarrow \sum_{i \in C} \frac{|D_{\text{valid},i}|}{N_{\text{valid}}} thr_i^{r+1}$
for each sample $x \in \mathcal{D}_{\text{test}}$ **do**
$\hat{x} \leftarrow p_\theta(q_\phi(x))$
$IS \leftarrow \|x - \hat{x}\|^2$
if $IS > thr_g^{r+1}$ **then**
Classify x as **Intrusion**
else
Classify x as **Benign**
end if
end for
If performance criteria : **break**
end for
return Final global model $(\boldsymbol{\theta}_g^{n_r}, \boldsymbol{\phi}_g^{n_r}, \boldsymbol{\Omega}_g^{n_r})$.

5 Experimental Setup and Results

5.1 Dataset Distribution

In the centralised setup, we used the 'network-wide' traffic from the 'benign' folder, which contains benign files from all nodes (analogous to a mirrored switch port), for training and validation. The data was split in an approximate 4:1 ratio for training and

validation (for threshold selection). The distribution of these data files among the clients was random and equitable.

5.2 Experimental Setup

The FL configuration consisted of 10 communication rounds. In each round, all clients were selected for training (fraction = 1.0 with four clients), with each performing $n_e = 3$ local epochs. The validation loss was used to determine the optimal hyperparameter combination. We tested proximal parameter λ in $\{10^{-1}, 10^{-2}\}$ learning rates lr of $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, and weight decays wd values of $\{10^{-3}, 10^{-4}, 10^{-5}\}$.

The framework is designed for computational efficiency, allowing the complete FL process to be trained and evaluated on a standard personal computer (8-core Intel Core i7-10870H CPU, 8 GiB RAM, GTX 1650). With the specified hyperparameter configuration, an end-to-end training cycle completes in approximately two hours. To further support deployment on resource-constrained systems, the framework incorporates memory-efficient techniques, such as processing labelled CSV files sequentially in chunks. This focus on efficiency extends to communication overhead; the model parameter matrices transmitted per client amount to approximately 30 kB, a volume that is orders of magnitude smaller than the local raw data, which can reach several gigabytes daily.

A Sigmoid activation function was applied to the final layer of all models to scale outputs to the input range of [0, 1]. The AE and VAE models used the ReLU activation function, while the AAE used LeakyReLU. For implementation, we used PyTorch [5] , NumPy, and Pandas libraries for feature engineering and backpropagation, and the Flower [6] for federated learning aggregation.

5.3 Performance Evaluation

We used four metrics to evaluate the binary (normal vs. attack) classification performance of the models, namely: Accuracy, Recall, False Discovery Rate (FDR), and F1-Score. The formulas rely on the following definitions:

- **True Positive (TP):** An attack is correctly identified as an attack.
- **True Negative (TN):** Benign traffic is correctly identified as benign.
- **False Positive (FP):** Benign traffic is incorrectly identified as an attack (a false alarm).
- **False Negative (FN):** An attack is incorrectly identified as benign (a missed detection).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{FDR} = \frac{FP}{TP + FP}$$

$$\text{F1-Score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

To establish a performance baseline for our proposed federated autoencoder, we benchmarked against two classical unsupervised anomaly detection algorithms implemented in a centralised manner: the Local Outlier Factor (LOF) and the One-Class Support Vector Machine (OC-SVM) [30]. Due to the computational intensity of these classical methods on the full dataset, we trained them on a randomly selected subset of nearly 10,000 normal instances. To ensure a fair and rigorous evaluation, all autoencoders were also benchmarked against the same downsampling method. We selectively present the results for the AAE on this subset alone, because it substantially outperforms the other autoencoder variants, thus providing a sufficiently strong point of comparison against the classical methods.

Our primary evaluation, presented in Table 2, compares our proposed methods against centralised models and the FL-ADS baseline [18] on a comprehensive test set encompassing all attack scenarios. The results show that our federated approach is highly effective, with the FedAvg AAE model achieving the highest accuracy (96.35%) among all unsupervised methods. While our model surpasses the accuracy of the supervised FL-ADS FedAvg baseline (93.68%), the supervised nature of FL-ADS gives it an inherent advantage in minimising false positives for known attacks, resulting in a lower FDR and a higher F1-score. In contrast, our unsupervised approach is designed to model the complex boundary of normal behaviour to detect any deviation. This strategy is paramount for identifying zero-day attacks and deliberately prioritises mitigating false negatives (high recall). The FedAvg AAE strikes the best balance among all unsupervised models, achieving the highest overall F1-score (79.12%) and the lowest FDR (29.95%) while maintaining excellent recall (90.93%), demonstrating its effectiveness in a realistic intrusion detection context.

A closer examination of the autoencoder variants across all tables reveals the architectural advantages of the AAE. As shown in Table 2, the AAE consistently outperforms the standard AE and VAE

[5] https://pytorch.org/
[6] https://flower.ai/

Table 2. Performance comparison of the centralised, proposed federated, and FL-ADS baseline approaches on a comprehensive test set comprising all attack scenarios

	Central						FedAvg			FedProx			FL-ADS	
Metric	OC-SVM	LOF	AE	VAE	AAE(ds)	AAE	AE	VAE	AAE	AE	VAE	AAE	Central	FedAvg
Accuracy	84.54	95.23	95.78	94.40	95.10	95.96	96.20	92.31	**96.35**	95.93	95.03	92.96	91.28	93.68
Recall	81.02	72.83	87.06	**90.98**	**92.00**	90.89	**90.93**	69.06	90.88	90.88	76.51	90.89	89.10	92.30
FDR	69.43	32.76	32.79	41.47	38.01	32.62	30.98	50.34	**29.95**	32.79	35.30	32.39	8.10	6.00
F1-score	44.39	69.92	75.86	71.23	74.07	77.39	78.47	57.77	**79.12**	77.28	70.11	77.54	90.40	93.10

Table 3. Performance evaluation for the external attacker scenario flows

	Central						FedAvg			FedProx		
Metric	OC-SVM	LOF	AE	VAE	AAE(ds)	AAE	AE	VAE	AAE	AE	VAE	AAE
Accuracy	93.85	74.99	98.04	98.42	98.89	99.62	99.68	91.55	**99.72**	99.49	89.32	99.51
Recall	99.77	16.64	94.26	99.75	**99.97**	99.75	**99.90**	74.71	99.74	99.75	65.76	99.75
FDR	0.17	0.08	0.97	4.88	**0.04**	1.02	0.97	4.33	0.70	1.44	2.92	1.38
F1-Score	90.53	28.17	96.59	97.38	98.16	99.36	99.46	83.91	**99.52**	99.15	78.41	99.18

Table 4. Performance evaluation for the compromised SCADA scenario flows

	Central						FedAvg			FedProx		
Metric	OC-SVM	LOF	AE	VAE	AAE(ds)	AAE	AE	VAE	AAE	AE	VAE	AAE
Accuracy	87.78	93.95	93.25	93.05	**94.16**	**93.63**	93.57	93.09	**93.60**	93.57	93.11	93.57
Recall	82.44	82.85	80.63	**81.87**	**83.75**	81.71	81.71	80.60	81.70	**81.71**	80.56	81.71
FDR	22.57	**3.99**	4.26	6.32	4.16	**4.00**	4.22	4.84	**4.07**	4.23	4.73	4.22
F1-Score	79.86	88.95	87.54	87.38	**89.39**	**88.28**	88.19	87.27	**88.25**	88.18	87.30	88.18

in terms of F1-score across all deployment scenarios—centralised, FedAvg, and FedProx. These results suggest that enforcing a prior distribution on the latent space via adversarial training produces a more regularised and discriminative representation of normal data than the statistical constraint of KL divergence used in VAEs. The VAE's comparatively lower performance, particularly its high FDR, can be attributed to its loss function, which may over-smooth the latent space, diminishing the model's ability to distinguish subtle anomalies. Furthermore, our deep learning models significantly outperformed the classical benchmarks; the FedAvg AAE's F1-score (79.12%), substantially higher than OC-SVM (44.39%) and LOF (69.92%).

Furthermore, the benefits of FL are evident, as our federated models frequently outperformed their centralised counterparts. For instance, in Table 2, the FedAvg AAE outperforms the Central AAE (79.12% vs. 77.39% F1-score), highlighting the strength of FL in building a more generalised global model. This trend is even more pronounced in the external attacker scenario (Table 3), where the FedAvg AAE achieves a near-perfect F1-score of 99.52%. The marginal performance difference between FedAvg and FedProx across the tables indicates that client drift was not a significant issue, likely due to our balanced data distribution. In such a homogeneous environment, the proximal term in FedProx, designed for stability, unnecessarily dampens the momentum of local updates.

Finally, the strong performance of the downsampled central model, AAE(ds), suggests that random downsampling acts as a powerful regularisation method. This superiority is evident not only in its high F1-score in the compromised SCADA scenario (Table 4) but also in its nuanced performance on specific attacks detailed in Table 5. While all models effectively detect high-volume attacks like 'BRUTE FORCE', the true benefit of downsampling emerges against subtle attacks like 'DELAY RESPONSE', where the recall (detection) rate dramatically increases from 46.51% to 93.02%.

Table 5. Recall performance of the AAE model per attack type, showing the mapping of specific attack scenarios to the 'Attack' class in the binary classification

Attack Type	Count	Dnsmpl	Central	FedAvg
BRUTE FORCE	393454	92.05	90.96	90.95
BASELINE REPL.	611	81.67	80.69	80.69
PAYLOAD INJ.	507	66.07	64.30	64.30
FRAME STACK.	505	98.22	86.93	81.98
QUERY FLOOD.	510	99.61	99.61	98.82
RECON	425	78.82	78.82	78.59
LENGTH MANIP.	470	87.45	87.02	87.02
DELAY RESP.	43	93.02	46.51	41.86

The underlying reason may be that training on the full dataset, dominated by millions of redundant, high-frequency traffic patterns, alters the training dynamic. Training on such a high absolute volume of dominant samples leads to overfitting, as the model becomes hyperspecialized in reconstructing—and thus memorising—these common patterns. Random downsampling effectively mitigates this by amplifying the learning signals from rarer normal variations, encouraging the model to learn a more robust representation of normalcy. However, this approach presents a clear tradeoff, as it discards information regarding the full benign data distribution. This impact is evident in our results (Table 2), where the downsampled model achieved a lower F1-score (74.07%) than the model trained on the full central dataset (77.39%).

6 Conclusion and future direction

In conclusion, this paper demonstrates that autoencoder variants are effective for detecting anomalous activities within CPSs. Specifically for the Modbus/TCP protocol, which is widely used in industrial settings, this approach can identify attackers originating from either external devices with forbidden access or internally compromised nodes. Furthermore, this detection capability, derived from normal daily network traffic, can be shared among distributed manufacturers via a federated framework with minimal performance loss, rendering it suitable for large or private datasets.

Future work could extend this research in several promising directions. First, the method's generalisability could be tested by evaluating it on other industrial network protocols. Second, alternative generative models warrant investigation; for example, a GRU-VAE could be explored within the federated setup to capture temporal dependencies in network flow sequences better. Finally, for scenarios with heterogeneous data distributions (e.g., those based on node IP addresses), other FL approaches such as Vertical FL or personalised FL could be investigated.

Acknowledgment

This work was supported by the Vice Chancellor for Research and Technology of Sharif University of Technology under Grant No. AI4032123.

References

[1] Rasim Alguliyev, Yadigar Imamverdiyev, and Lyudmila Sukhostat. Cyber-physical systems and their security issues. *Computers in Industry*, 100:212–223, 2018.

[2] Nicholas Jeffrey, Qing Tan, and José R Villar. A hybrid methodology for anomaly detection in cyber–physical systems. *Neurocomputing*, 568: 127068, 2024.

[3] Marie Baezner and Patrice Robin. Stuxnet. Technical report, ETH Zurich, 2017.

[4] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

[5] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[6] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[7] Lisa Liu, Gints Engelen, Timothy Lynar, Daryl Essam, and Wouter Joosen. Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 254–262. IEEE, 2022.

[8] Kwasi Boakye-Boateng, Ali A Ghorbani, and Arash Habibi Lashkari. Securing substations with trust, risk posture, and multi-agent systems: A comprehensive approach. In *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, pages 1–12. IEEE, 2023.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[10] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, Nadarajah Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, pages

756–767. IEEE, 2019.

[11] Beibei Li, Yuhao Wu, Jiarui Song, Rongxing Lu, Tao Li, and Liang Zhao. Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics*, 17(8):5615–5624, 2020.

[12] Syeda Aunanya Mahmud, Nazmul Islam, Zahidul Islam, Ziaur Rahman, and Sk Tanzir Mehedi. Privacy-preserving federated learning-based intrusion detection technique for cyber-physical systems. *Mathematics*, 12(20):3194, 2024.

[13] Viraaji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, 2021.

[14] Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, and Paulo Simões. Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process. In *International Conference on Critical Information Infrastructures Security*, pages 230–235. Springer, 2018.

[15] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414, 2016.

[16] Ons Aouedi, Kandaraj Piamrat, Guillaume Muller, and Kamal Singh. Federated semisupervised learning for attack detection in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 19(1):286–295, 2022.

[17] Ahmad Zainudin, Rubina Akter, Dong-Seong Kim, and Jae-Min Lee. Federated learning inspired low-complexity intrusion detection and classification technique for sdn-based industrial cps. *IEEE Transactions on Network and Service Management*, 20(3):2442–2459, 2023.

[18] Shaurya Purohit, Manimaran Govindarasu, and Benjamin Blakely. Fl-ads: Federated learning anomaly detection system for distributed energy resource networks. *IET Cyber-Physical Systems: Theory & Applications*, 10(1):e70001, 2025.

[19] Meryem Janati Idrissi, Hamza Alami, Abdelkader El Mahdaouy, Abdellah El Mekki, Soufiane Oualil, Zakaria Yartaoui, and Ismail Berrada. Fed-anids: Federated learning for anomaly-based network intrusion detection systems. *Expert Systems with Applications*, 234:121000, 2023.

[20] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1(2018):108–116, 2018.

[21] Inc. Acromag. Technical reference - modbus tcp/ip: Introduction to modbus tcp/ip. Technical report, Acromag, Inc., 2005. URL https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf.

[22] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. arxiv. *arXiv preprint arXiv:2003.05991*, pages 2593–2613, 2020.

[23] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[24] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[25] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[26] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6): 386–392, 2014.

[27] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

[28] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, 2012.

[29] Adrian Komadina, Mislav Martinić, Stjepan Groš, and Željka Mihajlović. Comparing threshold selection methods for network anomaly detection. *IEEE access*, 2024.

[30] Charu C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2nd edition, 2017. ISBN 978-3-319-47578-3.

Hamid Reza Dashtabadi earned his B.Sc. degree in Electrical Engineering from Amirkabir University of Technology in 2023, specialising in Telecommunications. He is currently completing his Master of Science degree in Electrical Engineering at Sharif University of Technology, focusing on Secure Telecommunications and Cryptography. His research interests include privacy in federated learning and Intrusion Detection methods for industrial protocols.

Siavash Ahmadi Siavash Ahmadi received the B.Sc., M.Sc. and Ph.D. degrees in Electrical Engineering from Sharif University of Technology, Tehran, Iran, in 2012, 2014 and 2020, respectively. He is currently an assistant professor at the Electronics Research Institute at the Sharif University of Technology. His particular fields of interest include Cryptology, Wireless, Cellular Networks, the Internet of Things and Artificial Intelligence, with an emphasis on security.