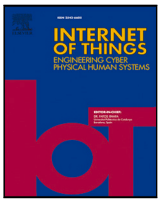
Internet of Things 28 (2024) 101340

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



DISFIDA: Distributed Self-Supervised Federated Intrusion Detection Algorithm with online learning for health Internet of Things and Internet of Vehicles ☆

Erol Gelenbe [a,b,c,*], Baran Can Gül [d], Mert Nakıp [a]

[a] Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Baltycka 5, Gliwice, 44–100, Poland
[b] Université Côte d'Azur, CNRS I3S, 06100, Nice, France
[c] King's College London, Department of Engineering, WC2R 2LS, London, UK
[d] Institute of Industrial Automation and Software Engineering, University of Stuttgart, Keplerstraße 7, Stuttgart, 70174, Germany

ARTICLE INFO

Keywords:
Federated learning
Confidential data
Distributed systems
Intrusion detection
Cybersecurity
Internet of Things (IoT)
Distributed Denial of Service (DDoS) attacks
Healthcare systems
Connected vehicles

ABSTRACT

Networked health systems are often the victims of cyberattacks with serious consequences for patients and healthcare costs, with the Internet of Things (IoT) being an additional prime target. In future systems we can imagine that the Internet of Vehicles (IoV) will also be used for conveying patients for diagnosis and treatment in an integrated manner. Thus the medical field poses very significant and specific challenges since even for a single patient, several providers may carry out tests or offer healthcare services, and may have distinct interconnected sub-contractors for services such as ambulances and connected cars, connected devices or temporary staff providers, that have distinct confidentiality requirements on top of possible commercial competition. On the other hand, these distinct entities can be subject to similar or coordinated attacks, and could benefit from each others' cybersecurity experience to better detect and mitigate cyberattacks. Thus the present work proposes a novel Distributed Self-Supervised Federated Intrusion Detection Algorithm (DISFIDA), with Online Self-Supervised Federated Learning, that uses Dense Random Neural Networks (DRNN). In DISFIDA learning data is private, and neuronal weights are shared among Federated partners. Each partner in DISFIDA combines its synaptic weights with those it receives other partners, with a preference for those weights that have closer numerical values to its own weights which it has learned on its own. DISFIDA is tested with three open-access datasets against five benchmark methods, for two relevant IoT healthcare applications: networks of devices (e.g., body sensors), and Connected Smart Vehicles (e.g., ambulances that transport patients). These tests show that the DISFIDA approach offers 100% True Positive Rate for attacks (one percentage point better than comparable state of the art methods which attain 99%) so that it does better at detecting attacks, with 99% True Negative Rate similar to state-of-the-art Federated Learning, for Distributed Denial of Service (DDoS) attacks.

☆ For the first and third author, this research was supported by the European Commission Horizon Europe – the Framework Programme for Research and Innovation (2021-2027) DOSS Project under Grant Agreement No: 101120270. All three authors made equal contributions to the research presented in this paper.

* Corresponding author at: Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Baltycka 5, Gliwice, 44–100, Poland.
E-mail addresses: gelenbe.erol@orange.fr (E. Gelenbe), baran-can.guel@ias.uni-stuttgart.de (B.C. Gül), mnakip@iitis.pl (M. Nakıp).

https://doi.org/10.1016/j.iot.2024.101340
Received 2 March 2024; Received in revised form 8 July 2024; Accepted 21 August 2024
Available online 4 September 2024

1. Introduction

The healthcare of a patient can involve tens of distinct Internet connected entities [1], ranging from the general practitioner, the medical test laboratories, the specialists, the wearable body sensors, imaging devices and systems, ambulances and other connected vehicles, and the security and administrative services of healthcare providers such as hospitals, clinics, ambulances, emergency services, and insurance companies. Since in the course of our discussion we will introduce, and then use, several acronyms, to discuss this area, its cybersecurity needs, and the related attack detection methods, the acronyms that we use are summarized in Table 1.

The complexity of such systems is growing with the availability of connected body sensors, imaging sensors and other devices, as well as the Internet of Vehicles (IoV) for ambulances and connected vehicles that carry patients in normal and emergency circumstances, both for hospitals and in ambulatory care. The explosion of the number of such connected devices, just for the Internet of Things (IoT) digitally traceable medical devices, is illustrated by a market size that is expected to exceed $400 billion by 2025 [2]. Unfortunately, at the same time, public and private health service organizations are becoming a significant target for cyber criminals [3] with hackers that may be motivated to exploit sensitive patient data, demand ransoms, or attack health systems for personal or political reasons, even by trying to attack devices implanted in the body of patients [4].

Indeed, while the development of wearable or implantable medical devices (IMDs) with lightweight communication protocols have helped to simplify and improve healthcare, and connected vehicles such as ambulances can improve the timely operation of such complex systems, they also pave the way for major risks of cyberattacks [5,6].

Among these, Denial of Service (DoS) and Distributed DoS (DDoS) are the most common, and are often also part of more sophisticated attacks. They overload their victim devices with large flows of communications over a short time [7], resulting in operational slowdowns and delays, which can lead to system shutdowns. DDoS attacks typically exploit Internet of Things (IoT) devices that are compromised to generate further attacks. Over the year 2022, the number of DDoS attacks was reported to have increased by 109% at the network layer, and by 72% at the application layer [8].

Due to the prevalence of such attacks, Intrusion Detection Systems (IDSs) that detect cyberattacks, and particularly DoS and DDoS, are essential components of cybersecurity.

2. Related work

Much work on the design and evaluation of different IDSs has been conducted using Machine Learning (ML), because of the high accuracy levels and low false alarm rates, that this approach has attained for the detection of threats in network traffic [9–11], and for detecting different attack types with a single iDS [12–14].

Traditional ML-based IDSs learn from large datasets that are typically collected during real attacks, as well as in the presence of benign traffic. However in many healthcare systems [15–17] and for autonomous vehicles [18], multiple distinct entities must operate together and collaborate in a data-driven environment [19–21], where each device and vehicle receives a distinct flow of network traffic. In such distributed collaborating devices or systems, intrusion detection methods can benefit from Federated Learning (FL) [22,23], which allows the exchange of experience learned by each participant, without the need to actually transfer and share the data or traffic streams received by the different participants.

In FL or collaborative learning, a set of N collaborating entities, such as IoT devices, vehicles, servers or nodes, use a common ML model with identical parameters (e.g. same neural network architecture with its specific synaptic weights), although the actual value of the parameters may vary from one entity to the other. Each entity receives a distinct dataset, such as a sequence of network packets, which it uses to learn its own parameter values, without sharing its data with other entities. However it does share its weights with other entities, so that each entity profits from the experience of the other ones. In addition to the fact that FL does not share large masses of packets or datasets between collaborating entities, FL also respects the privacy and confidentiality requirements imposed by many applications such as digital health.

Thus in FL, after some learning time, or number of learning steps, or quantity of data used for learning, each entity can share its internal state or synaptic weights, with all other $N-1$ entities with which it is cooperating. After receiving the parameters from the other entities, each individual entity updates its own parameter set based on the its own parameters and the parameters received from the other entities. We will say that this scheme is **Synchronous Distributed** because the distinct entities share their parameter sets, for a total of $N(N-1)$ transfers, after a predetermined amount of learning that can be fixed, either as an amount of time or as an amount of learning data, and then each entity carries its own parameter update locally, in a distributed manner.

To avoid having to exchange the parameters with $N(N-1)$ data transfers between all entities, some FL methods will require that each entity forwards its parameters to the same designated entity or a common server, and that server will carry out a common update of the parameters and then send them back to the other entities, requiring at most $2N$ (or $2N(N-1)$ if one of the entities behaves as the server) transfers of parameters, (or $2N(N-1)$ if one of the entities behaves as the server). We call this scheme the **Synchronous Centralized** approach, since the common parameter update are carried out in a single location.

While many traditional forms of FL operate under one of these two Synchronous schemes, this approach can be impractical for several reasons when the entities are network or server nodes, and the datasets are composed of network IP packets, as in the field of Network Cybersecurity. Firstly, the number of packets received by different nodes will vary widely over most time periods, because nodes will receive and send packets based on the specific needs of their individual end users and of the applications they support. Thus fixing a predetermined number of packets, and then updating all the weights and sharing them at the same time, implies that the different nodes may stop and try to send their parameters at very different time intervals, and the nodes with less traffic will create lengthy waiting periods for the other nodes. Alternatively, if the time intervals for learning are the same for the different nodes, then the amount of learning that is accomplished will vary significantly between the different nodes since in a given time epoch different nodes may be processing very different numbers of packets.

Table 1
List of Abbreviations in Order of Appearance.

Abbreviation	Definition
IoV	Internet of Vehicles
IMDs	Implantable Medical Devices
DoS	Denial of Service
DDoS	Distributed Denial of Service
IoT	Internet of Things
IDS	Intrusion Detection Systems
ML	Machine Learning
FL	Federated Learning
DISFIDA	Distributed Self-Supervised Federated Intrusion Detection Algorithm
IDAF	Intrusion Detector with Adaptive Federation
CAN	Controller Area Network
AWID	Aegean Wi-Fi Intrusion Dataset
GAN	Generative Adversarial Network
MitM	Man-in-the-Middle
ITS	Intelligent Transportation Systems (ITS)
DRNN	Deep Random Neural Network
SWBC	Statistical Whisker-based Benign Classifier
LLA	Local Learning Algorithm
DAFU	Decentralized Asynchronous Federated Update
ACN	Average with Closest Node
ACN-L	Average with Closest Node per Layer
DOF-ID	Decentralized and Online Federated Learning Intrusion Detection
RAM	Random-Access Memory
HTTP	Hypertext Transfer Protocol
TNR	True Negative Rate
TPR	True Positive Rate
FHE	Fully Homomorphic Encryption
TCP	Transmission Control Protocol
IP	Internet Protocol
KB	Kilobyte
MB	Megabyte
AI	Artificial Intelligence

2.1. Related work

We now review related work that can take a centralized or decentralized approach to FL-based IDS. While the centralized FL approach updates all the ML parameters in a single server that builds the FL model, in decentralized FL we have individual collaborating nodes that exchange their parameters and allow each local system to update its parameters accordingly. .

In recent work [24], experiments involving DDoS and False Data Injection attacks regarding vehicular cyber–physical systems were conducted. The authors subsequently introduced a secure FL approach that employs Elliptic Curve Cryptography and Asynchronous FL techniques.

In [25], Federated Deep Belief Network-based IDS was developed to detect intrusions in wireless networks, and tested using the Aegean Wi-Fi Intrusion Dataset (AWID). In [26] a cross-layer federated learning system with K-means clustering for intrusion detection in lightweight IoT devices, while in [27], the impact of attack parameters of data poisoning attacks were experimentally evaluated with the UNSW-NB15 dataset. In [28], a centralized federated architecture was designed for malware detection regarding the industrial IoT using Generative Adversarial Network (GANs). While this architecture demonstrates impressive attack detection accuracy, it relies on the availability of a validation set at the server, which could potentially compromise the privacy of user data. The work in [29] presents a multi-step prediction method for DDoS attacks with Hidden Markov Models within a centralized FL architecture that incorporates Reinforcement Learning. Meanwhile a multi-class classifier for FL-based IDS was developed in [30] for various data distributions, and a self-learning distributed approach for detecting compromised IoT devices affected by the Mirai malware was discussed in [31]. An anomaly detection method based on centralized FL for classifying and identifying attacks in the IoT is discussed in [32], and tested using a dataset that includes Man-in-the-Middle (MitM) and flood attacks. An architecture to mitigate and respond to DDoS attacks against the industrial IoT is proposed in [33].

2.2. Decentralized federated learning

Safeguards against gradient attacks were addressed in [34], which introduced a decentralized FL framework, leveraging a peer-to-peer network to facilitate the transmission, aggregation, and synchronization of local models. In [35], decentralized FL was employed to identify anomalies in packet traffic generated by IoT devices. This approach involves sharing all federated IDSs with each unique participant to calculate a weighted average. In contrast, the work in [36] utilized blockchain-based FL within a decentralized architecture, focusing on the detection of poisoning attacks.

Another study [35] uses decentralized FL to detect anomalies in communications exchanged by IoT devices when all federated IDSs use the average of all the weights learned by each device. In [36], a blockchain-based FL was used to detect poisoning attacks.

Table 2
List of Mathematical Symbols in Order of Appearance (1).

Symbol	Definition
N	Number of collaborating nodes
n	Always refers to node n or DISFA-n
l	Always refers to time window l
W_n	Neural weights for collaborating node n
p_n^l	Packet arriving to node n at time t
T_n	Length of time window for node n
P_n^l	Set of packets arriving at node n in window l
F^*	Fusion rule of model weights
W_n^*	Local weights of node n
x_n^l	Input vector at each window
y_n^l	Binary intrusion decision at each window l
C_{Max}	Max. datarate of incoming comms at nodes
P_{Max}^n	Max. number of packets arriving to node n
L	Maximum length of packets in bytes
μ_n^l	Normalized average length of the packets
λ_n^l	Normalized average arriving datarate in packets/sec
ρ_n^l	Normalized average number of bytes/sec
H	Number of cluster of neurons
$\Psi(\Lambda)$	Activation function of a hidden layer neuron cluster
Λ	Input of each neuron in a cluster
p	Probability of a neural trigger in a neuronal cluster
λ^+, λ^-	Rates of external *excitatory and inhibitory* input spikes
$\hat{x}_{(n,h)}^l$	Output vector of each layer h
$W_{(n,h)}^l$	Connection weight matrix from layer $h-1$ to h
$z_{n,i}^l$	Abs. diff. between actual and predicted value of statistic i

In addition, FL that is managed with blockchain was suggested [37] for intrusion detection, with various algorithms to compute the neural weights, such as Scaled Conjugate Gradient, Bayesian Regularization, and Levenberg–Marquardt.

In [38], the lightweight, decentralized deep learning architecture called FED-IDS, was introduced to detect cyberattacks in heterogeneous Intelligent Transportation Systems (ITS). Although itd performance is promising, the resulting performance does not meet the real-time requirement of ITS as it is not computationally efficient. Other recent work [39], presents a detection model for a mesh satellite network architecture and asynchronous FL to identify cyberattacks in autonomous vehicles, and enhances its trust in connected nodes using a blockchain.

The single point of failure problem is addressed in [40] for decentralized FL and distributed ledger techniques, and the paper identifies open problems for future research directions. A strategy named GWOFL for resilient FL based on grey wolf optimization [41], addresses communication challenges in IoT-driven industry 5.0 contexts, including slow convergence, communication overhead, and susceptibility to adversarial attacks. Personalized FL algorithms for intrusion detection in IoT devices with data heterogeneity limited to a few applications are discussed in [42]. In [43], an effective decentralized FL framework known as DeProFL uses insight from various IoT systems, and tries out different data sets and models to enhance the learning system's performance.

3. The DISFIDA system proposed in this work

In the present paper we propose the novel Distributed Self-Supervised Federated Intrusion Detection Algorithm (DISFIDA) that coordinates learning among N distinct nodes. At each node, DISFIDA uses an instantiation of the same Intrusion Detection Algorithm with Federation (IDAF) that learns locally, provides intrusion detection, and updates its neural weights W_n asynchronously from the other nodes. Each IDAF uses exactly the same neural network architecture with the same set neural network weights. The mathematical symbols that are used in this paper are summarized in Table 2 and Table 3

Locally, $IDAF-n$, $1 \leq n \leq N$, receives the packet p_n^t at time t, whose size in bytes is $|p_n^t|$. $IDAF-n$ runs its local learning algorithm during successive time windows of length T_n (fixed for node n), using P_n^l, the set of packets that arrive at node n within the lth time window.

It therefore learns during each successive the window of T_n, by updating its set of W_n^* weights in an auto-associative manner, to compute the new locally learned value W_n, from the arriving packets P_n^l using the non-attack decisions. Specifically, if $IDAF-n$ does NOT detect an attack over these packets, then it modifies its local weights so that its output matches its input as closely as possible, resulting in the new local weights W_n. Auto-associative learning does not occur from ATTACK decisions, so that $IDAF-n$ is only learning "normal benign traffic".

Of course $IDAF-n$ outputs both attack and non-attack decisions based on all the packets (attack and non-attack) that it receives. Furthermore, all the performance evaluations results that we present are for *ALL* the attack and non-attack traffic received by all $IDAF-n$ nodes, $1 \leq n \leq N$.

Once the update is complete, $IDAF-n$ shares its weights W_n with all the other $N-1$ nodes in an Asynchronous and Decentralized manner, **without sharing the packets that it receives with other nodes**. DISFIDA also provides a fusion rule F to all the N local IDAFs, so that – after learning locally – each $IDAF-n$ also updates its own weights to a new value W_n^* by using F and the weights it has received from other nodes. In summary:

Table 3
List of Mathematical Symbols in Order of Appearance (2).

Symbol	Definition
θ_n^l	Decision threshold at window l of node n
I_n^l	Parameters of each local $IDAF-n$ at window l
$\mathcal{D}_n^l$	Set of window indices at window l
Γ_n^l	Measure of trust at window l of node n
K	Number of packets for intrusion decision
γ	Decision threshold in SSL
$\mathcal{R}_n^l$	Set of nodes whose weight updates were received
G_n^l	Generalization ability of $IDAS-n$
$M_{(n,l)}^O$	Total packet length received until the end of window l
$M_{(n,l)}^L$	Length of all packets in the time windows l, used for learning
$\Lambda_{(n,l)}^O$	Mean inter-transmission times for all time widows till the lth
$\Lambda_{(n,l)}^L$	Mean inter-transmission times used for learning
Δ_n^l	Average value of data adequacy at window l
κ_n^l	Expected performance in window l
W_R	$(H \times H)$ matrix with randomly generated elements with uniform distribution in $[0,1]$
$u_{n,i}^l$	Upper whisker for ith input
$Q_{n,i}^L$	Lower quartile for ith input
$Q_{n,i}^U$	Upper quartile for ith input
ζ_n^l	Average number of abnormal statistics
$\mathcal{M}_n^l$	Parameters received from the federated nodes at window l
$I_m^l(x_n^k)$	Decision of the IDS of node m for the input x_n^ka
C_n^l	Set of concurring nodes
$m_l(n)$	Node at time window l

- Each IDAF has the same feedforward neural network structure with the same number of neurons, and the same number of neuronal weights.
- The N IDAFs operate asynchronously with respect to each other, since each of them can have a distinct value of T_n.
- Each IDAF will learn distinct weight values based on the traffic it receives locally, and then it will "fuse" its own weights with the weights from different neighbors, using the DISFIDA fusion rule F^* discussed in Section 5.
- IDAFs detect attacks locally using its weights W_n for packet traffic arriving at their respective local node. The detection process is carried out in successive windows of length T_n.
- When $IDAF-n$ receives traffic packets at node n, it first **detects whether this is an attack or not** using its current weight set W_n, and then it locally updates its internal parameters based on the same local self-supervised on-line algorithm that is used by each IDAF to obtain W_n^*. This update occurs after each window. Once the update is complete it shares its weights with all other $N-1$ nodes. Each $IDAF-n$ then fuses its weights using the rule F^*.

DISFIDA coordinates the IDAFs by providing to all of them, the following elements:

- The time window T_n which determines the number of packets that each IDAF uses for learning, and the frequency with which the IDAF weights are updated.
- The Fusion Rule F^* that is used to update the current set of weights W_n^* for node $IDAF-n$, with the set of weights W_v^* it has most recently been received from other nodes $v \neq n$.
- When $IDAF-n$ receives the weights from a majority $V > (N-1)/2$ of other nodes, it produces the new value of the weights W_n at node n, using the fusion rule F^*, asynchronously with other nodes.

While taking the average value of **all** the neural weights is commonly used in many FL approaches, DISFIDA obviously does not do this. Thus, in this paper we will compare several approaches for fusing the weights – including DISFIDA – as discussed in Section 5, and evaluate their resulting accuracy with regard to intrusion detection. In particular, we observe that **simple averaging can lead to results that are less accurate** than other approaches.

In the sequel, we detail DISFIDA, and evaluate its performance for the detection of DoS and DDoS attacks, in two different scenarios of a Collaborating IoT Network, and Connected Smart Vehicles. In the first scenario with IoT devices, we examine three different types of cyberattacks from two well-known public datasets: Kitsune [44,45] and Bot-IoT [46]. For the Smart Vehicle scenario, we use the Automotive Controller Area Network (CAN)-bus intrusion dataset [47] that relates to six distinct nodes ($N = 6$) for our evaluation. For both scenarios, we also compare the performance of DISFIDA with five other benchmark methods.

Table 4
Comparison of DISFIDA with other Best in Class Methods. The references from which the results are taken are shown explicitly in this Table. Note that all the numerical results and Datasets are as stated by the authors of the referenced papers.

Method	Dataset	TPR	TNR	Accuracy
FEDDBN-IDS [25]	Aegean WiFi Intrusion	0.99	0.993	0.992
NIDEFML [37]	NST-KDD	0.987	0.991	0.989
Synchronous [48]	Kitsune MIRAI	1	0.97	0.98
Synchronous [48]	DOS HTTP	0.99	0.91	0.93
Synchronous [48]	DDOSS HTTP	0.92	0.86	0.88
DISFIDA	Kitsune MIRAI	1	0.99	0.99

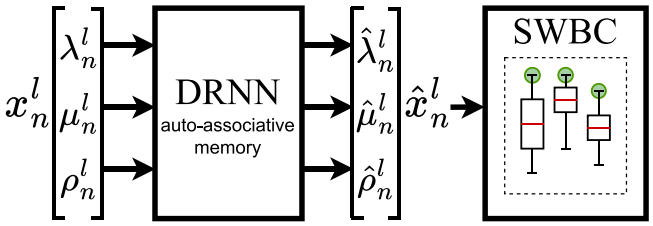


Fig. 1. Computation that is carried out by the IDAF at node n during window l that we denote by I_n^l.

The results show that the DISFIDA provides significant performance gains compared to learning from local data alone, and that it outperforms the benchmark methods. Meanwhile, due to the timely updates provided by DISFIDA's self-supervised learning, we observe that it operates efficiently with under 0.03% communication overhead when no additional data encryption is used.

The comparison of DISFIDA with the other "best in class" methods [25,37,48], is summarized in Table 4. The references from which the results are taken are shown explicitly in this Table, and that all the numerical results and Datasets are as stated by the authors of the referenced papers. These results show that DISFIDA offers 100% True Positive Rate (TPR) for attack detection, equal to the Synchronous method [48] on the same Kitsune MIRAI Dataset, but is better than the other effective methods by 1%. DISFIDA offers a True Negative Rate (TNR) of 99% which is comparable to two of the other best in class methods, and clearly better than the third one. Similarly, the Accuracy metric is identical – within two decimal digits – of other two methods, but better than Synchronous. We note that one of these other methods [37] is an online report, which has not yet been published after peer review.

The rest of this paper is organized as follows. Section 3.1 details the IDS used in this paper, that we name the Intrusion Detector with Adaptive Federation (IDAF). Section 4 presents DISFIDA, and its self-supervised local learning and distributed asynchronous federated updates. Section 5 evaluates the performance of DISFIDA and compares it against other methods. Finally, Section 6 presents conclusions and highlights future research directions.

3.1. IDAF: DISFIDA's intrusion detection algorithm

In Fig. 1 we show $IDAF-n$, the Intrusion Detection system used by DISFIDA for each node n. It consists of a Deep Random Neural Network (DRNN) [49], with the Statistical Whisker-based Benign Classifier (SWBC), that provides a binary intrusion decision y_n^l based on the input vector x_n^l, at each window l. A "window" is a time interval of length T_n secs, that is used to learn from incoming packets, and provide a decision about whether an attack is in progress.

Let p_n^t be a packet arriving to node n at time t, its size in bytes is $|p_n^t|$, and P_n^l is the set of packets that arrive at node n within the lth time window:

$$P_n^l = \{p_n^t : \ (l-1)\,T_n \le t < l\,T_n\}. \tag{1}$$

$|P_n^l|$ is the total number of packets that arrive to node n during window l, and we know that this number is upper bounded by some value P_{Max}^n due to the maximum capacity C_{Max}, in bytes per second, of the incoming communication channel to node n, i.e. $0 \le |P_n^l| \le P_{Max}^n$. Furthermore all packets have a maximum length of L in bytes. Indeed, we know that $P_{Max}^n \le C_{Max}/L$, but in some cases it may be strictly smaller due to specificities of the communication protocol.

The input to $IDAF-n$ during the lth time window, is a vector $x_n^l = [\mu_n^l, \lambda_n^l, \rho_n^l]$ composed of three metrics which are normalized between 0 and 1, namely:

- μ_n^l the normalized average length of the packets arriving in window l,
- λ_n^l the normalized average number of packets per second,
- ρ_n^l the normalized average number of bytes per second.

They are formally defined as:

$$\mu_n^l = \frac{\sum_{p\in P_n^l}|p|}{L\times|P_n^l|}, \quad \lambda_n^l = \frac{|P_n^l|}{P_{Max}^n\times T_n}, \quad \rho_n^l = \frac{\sum_{p\in P_n^l}|p|}{L\times T_n}, \tag{2}$$

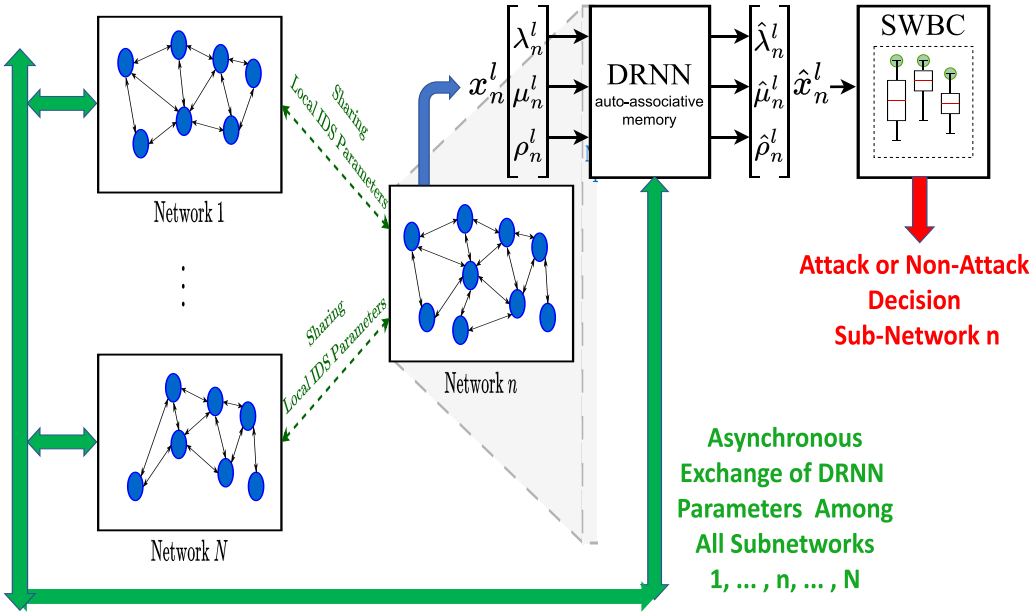


Fig. 2. System representation of DISFIDA and its internal interconnections.

The $IDAF-n$ Input-and-Output, including the DRNN [49] and the SWBC (Statistical Whisker-based Benign Classifier) are shown in Fig. 1. Note that during Auto-Associative Learning for the DRNN discussed in Section 3.2, the DRNN weights are chosen by **minimizing the difference** between the output vector shown in Fig. 1, as is done in [50]:

$$\hat{x}_n^l = [\hat{\mu}_n^l, \hat{\lambda}_n^l, \hat{\rho}_n^l], \tag{3}$$

and the input vector $x_n^l = [\mu_n^l, \lambda_n^l, \rho_n^l]$.

3.2. Deep Random Neural Network (DRNN)

$IDAF-n$ is a DRNN [49] with a square structure of H fully connected layers, each with H clusters of neurons. As the number of DRNN inputs (i.e. the number of traffic statistics) equals 3, we set $H = 3$with two hidden layers, and one output layer comprised of linear neurons. Each hidden layer neuronal cluster uses the DRNN's activation function [49]:

$$\Psi(\Lambda) = \frac{p\,(r+\lambda^+)+\lambda^-+\Lambda}{2\,[\lambda^-+\Lambda]} - \sqrt{\left(\frac{p\,(r+\lambda^+)+\lambda^-+\Lambda}{2\,[\lambda^-+\Lambda]}\right)^2 - \frac{\lambda^+}{\lambda^-+\Lambda}}, \tag{4}$$

where Λ is the input of each neuron in a cluster, p is the probability of a neural trigger in each cluster of neurons of each layer, and λ^+ and λ^- are the rates of external *excitatory and inhibitory* Poisson-distributed input spikes. This DRNN estimates the vector of expected statistics values, denoted by $\hat{x}_n^l$, based on the input x_n^l:

$$\hat{x}_{(n,1)}^l = \Psi([x_n^l, 1]\, W_{(n,1)}^l), \tag{5}$$

$$\hat{x}_{(n,h)}^l = \Psi([\hat{x}_{(n,h-1)}, 1]\, W_{(n,h)}^l), h = 2, \ldots, H-1, \tag{6}$$

$$\hat{x}_n^l = [\hat{x}_{(n,h-1)}^l, 1]\, W_{(n,H)}^l, \tag{7}$$

where the output vector of each layer h for node n in window l is denoted by $\hat{x}_{(n,h)}^l$. The connection weight matrix between layers $h-1$ and h is denoted by $W_{(n,h)}^l$. In addition, 1 is concatenated to the input of each layer h as a multiplier of the bias, represented by $[x_n^l, 1]$ or $[\hat{x}_n^l, 1]$.

3.2.1. Statistical Whisker-based Benign Classifier (SWBC)

SWBC, originally developed in [14], is used to make decisions in $IDAF-n$, the absolute difference between the actual and predicted values of each statistic:

$$z_{n,i}^l = |x_{n,i}^l - \hat{x}_{n,i}^l|, \tag{8}$$

where $x^l_{n,i}$, $\hat{x}^l_{n,i}$ are, respectively, the i–th elements of the vectors x^l_n, $\hat{x}_{n,i}$ defined earlier.

Then, it computes the number of statistics whose values are significantly different from their predicted values:

$$\zeta^l_n = \sum_{i\in\{1,2,3\}} \mathbf{1}[\ |x^l_{n,i} - \hat{x}^l_{n,i}| > \frac{3}{2} w^l_{n,i}\ ] \tag{9}$$

using the statistical whisker values $\{w^l_{n,i}\}_{i\in\{1,2,3\}}$. Finally, SWBC provides the binary intrusion decision based on the decision thresholds θ^l_n for $1 \le i \le 3$:

$$y^l_n = \mathbf{1}[\zeta^l_n > \theta^l_n], \tag{10}$$

Notice that SWBC uses the decision parameters $\{w^l_{n,i}\}$ and θ^l_n for $i \in \{1, 2, 3\}$ which are set during the learning of all the parameters, and does not use any other parameter setting or decision threshold adjustment.

4. DISFIDA: Distributed Federated Intrusion Detection Algorithm

We now present DISFIDA that coordinates the collaboration between the N IDAFs, to enhance their ability to detect attacks using the federated learning framework, where each $IDAF - n$ learns locally from the network traffic that it receives, in a self-supervised manner, and shares its experience by forwarding its weights to the other $N - 1$ IDAFs, as shown schematically in Fig. 2, sharing knowledge with all collaborating nodes to improve overall security, while confidentiality of local traffic is preserved at each node n.

4.1. DISFIDA and its pseudo-code

The DISFIDA Pseudo-Code is given in **Algorithm 1**.

Algorithm 1 The DISFIDA Pseudo-Code

$\mathcal{R}_n = \{\}$
ParComp(**Federated_Update**())
for all $l \ge 1$ **do**
  SSL()
  Broadcast(**Encrypt**(I^l_n)) ▷ 4. & 5.
end for
function SSL
  if $\frac{1}{K}\sum_{k=l-K}^{l-1} y^k_n > \gamma$ **then**
    $\mathcal{D}^l_n = \mathcal{D}^{l-1}_n \setminus \{k\}_{k\in\{l-K,\ldots,l-2\}}$
  else if $y^{l-1}_n == 0$ **then**
    $\mathcal{D}^{l-1}_n \cup \{l-1\}$
  else
    $\mathcal{D}^{l-1}_n$
  end if
  Γ^l_n = **ComputeTrust**(I^l_n)
  if $\Gamma^{l-1}_n < \Theta$ **and** $\frac{1}{K}\sum_{k=l-K}^{l-1} y^k_n \le \gamma$ **then**
    I^l_n = **LLA**(D^l_n)
  else
    $I^l_n = I^{l-1}_n$
  end if
end function
function Federated_Update
  when updateReceived **from** m **then**
    $\mathcal{R}_n \leftarrow \mathcal{R}_n \cup m$
    if $|\mathcal{R}_n| \ge (N-1)/2$ **then**
      $I^l_n \leftarrow F^*(\mathcal{D}^l_n, \{I^l_m\}_{m\in\mathcal{R}_n})$
      $\mathcal{R}_n \leftarrow \{\}$
    end if
  end when
end function

For each node n, DISFIDA operates over successive time windows of length T_n seconds. Due to its asynchronous nature, DISFIDA does not require synchronization among different collaborating nodes, and each node n may launch DISFIDA (i.e. the first time window $l = 1$) at any locally chosen time.

At the beginning of window l, first, the network traffic of node n that was received during the previous window $l-1$, is analyzed by I_n^l to identify any existing malicious traffic. To this end, $IDAF-n$ estimates the probability of intrusion, y_n^{l-1}, for the traffic observed during window $l-1$, as described in Section 3.1. The parameters of each l$IDAF-n$, denoted by I_n^l, use the DRNN [49] which is updated based on local *benign* network traffic, and it is then updated with the weights that are received from the collaborating nodes in DISFIDA. During this procedure, DISFIDA requires neither human intervention, not data labeling.

As shown on Line 2 of this Algorithm, DISFIDA runs the distributed asynchronous federation algorithm in parallel to the local training of I_n^l and broadcasting the encrypted IDS parameters for each window l between Lines 3–6. Within each window l, the local self-supervised learning algorithm (SSL) between Line 7 and Line 21, is called at Line 4. Between Line 8 and 14, SSL updates the set of window indices $\mathcal{D}_n^l$ that contains normal traffic with respect to IDS's decision y_n^k. In Line 15, the trust metric Γ_n^l in I_n^l is computed as in Section 4.2.

The parameters of I_n^l are then updated between Lines 16–20 using the local traffic at node n via the Local Learning Algorithm (LLA), where LLA is an unsupervised auto-associative learning algorithm that trains each $IDAF-n$ on locally collected benign traffic, the LLA does not require any malicious traffic or external data collection for training.

In parallel with the SSL and broadcasting, the "Federated_Update" given in Lines 24–30 runs continuously. Unless this distributed computation is stopped, this algorithm will continue operating either by waiting for an update on Line 23, or executing the code in Lines 24–29 when an update is received from another collaborating node m. In Line 24, the set of nodes $\mathcal{R}_n$, whose weight updates have been received, is updated. In Line 26, if updates are received from the majority of nodes, the parameters I_n^l are updated using F^* as indicated in Section 4.4. Then, in Line 27, one empties the set $\mathcal{R}_n$.

Note that the time window T_n of DISFIDA at node n should obviously be at least equal to the total computation times needed for intrusion detection, local learning, and sending federated update using F^*. DISFIDA does not require synchronization for the time windows, or for collaborating nodes. Hence, each node n may launch DISFIDA (i.e. the first time window $l = 1$ may start) at any time.

At the beginning of window l, first, the network traffic of node n collected in the previous window $l-1$ is analyzed by I_n^l to identify any existing malicious traffic. Subsequently, the parameters of each local $IDAF-n$, which are denoted by I_n^l, are using the DRNN [49] updated first based on local *benign* network traffic, and secondly using the updates shared by the collaborating nodes in DISFIDA. One should note that during this procedure, DISFIDA requires *no human intervention* or *no data labeling*.

At the beginning of each window l, $IDAF-n$ estimates the probability of intrusion, y_n^{l-1}, for the traffic observed during that window, as described in Section 3.1. Based on the decision variable y_n^{l-1}, the following steps are followed as part of the self-supervised learning framework:

1. The set of window indices $\mathcal{D}_n^l$, that have been identified to contain normal traffic, is updated:

$$\mathcal{D}_n^l = \mathcal{D}_n^{l-1} \setminus \{k\}_{k\in\{l-K,\ldots,l-2\}}, \textbf{if } \frac{1}{K}\sum_{k=l-K}^{l-1} y_n^k > \gamma \tag{11}$$

$$\mathcal{D}_n^{l-1} \cup \{l-1\}, \textbf{else if } y_n^{l-1} = 0 \tag{12}$$

$$\mathcal{D}_n^{l-1}, \textbf{otherwise}. \tag{13}$$

2. The measure of trust Γ_n^l in the decisions of I_n^l, is computed, as in Section 4.2.
3. The parameters of I_n^l are then updated using the local traffic at node n via the Local Learning Algorithm (LLA):

$$I_n^l = \text{LLA}(\mathcal{D}_n^l), \textbf{if} \quad \Gamma_n^{l-1} < \Theta \ \textbf{ and } \ \frac{1}{K}\sum_{k=l-K}^{l-1} y_n^k \le \gamma \tag{14}$$

$$I_n^{l-1} \ \textbf{otherwise}, \tag{15}$$

   where LLA is an unsupervised auto-associative learning algorithm that trains each $IDAF-n$ on locally collected benign traffic. The LLA does not require any malicious traffic or external data collection for training.
4. After a given node n updates its local parameters, it forwards I_n^l to the other $N-1$ nodes in DISFIDA.
5. For some implementations of DISFIDA, the shared data may also be stored in a common shared memory.
6. As shown in Fig. 2, in parallel to the local training of I_n^l in a window l, each node n, waits to receive the up-to-date IDS parameters from $V > (N-1)/2$ of the other nodes in DISFIDA, then the parameters of the local IDAF I_n^l are updated using F^* which is DISFIDA's distributed asynchronous federation algorithm:

$$I_n^l \leftarrow F^*(\mathcal{D}_n^l, \{I_m^l\}_{m\in\mathcal{R}_n^l}), \tag{16}$$

   where $\mathcal{R}_n^l$ is the set of nodes whose weight updates have been received in the l-the step at node n.

The time window T_n of DISFIDA T_n at node n should obviously be at least equal to total computation times for intrusion detection, local learning, and making federated update using F^*.

4.2. Trust estimation for self-supervised learning

The $IDAF-n$ local self-supervised learning is based on our previous work [51], requiring no human intervention or prior data collection. The Trust parameter is estimated using (14) from the traffic data it learned until window l:

$$\Gamma_n^l = R_n^l\, G_n^l \tag{17}$$

where R_n^l and G_n^l characterize the representativeness of the learned traffic and the generalization ability of $IDAS-n$.

4.2.1. Representativeness of the traffic learned

The representativeness R_n^l of the traffic learned through the KL-Divergence [52]. Using $1/\Lambda_{(n,l)}^O$ and $1/M_{(n,l)}^O$, respectively, as the mean inter-transmission times and of the length of all packets in all time windows until the end of l, while $1/\Lambda_{(n,l)}^L$ and $1/M_{(n,l)}^L$ are respectively the means of inter-transmission times and lengths of all packets in the time windows that are used for learning, the values $\Lambda_{(n,l)}^L$, $\Lambda_{(n,l)}^O$, $M_{(n,l)}^L$, and $M_{(n,l)}^O$ are obtained from:

$$\Lambda_{(n,l)}^L = \frac{1}{|\mathcal{D}_n^l|}\sum_{k\in\mathcal{D}_n^l}\lambda_n^l, \quad \text{and} \quad \Lambda_{(n,l)}^O = \frac{1}{l}\sum_{k=1}^{l}\lambda_n^l, \tag{18}$$

$$M_{(n,l)}^L = \frac{1}{\frac{1}{|\mathcal{D}_n^l|}\sum_{k\in\mathcal{D}_n^l}\mu_n^l}, \quad \text{and} \quad M_{(n,l)}^O = \frac{1}{\frac{1}{l}\sum_{k=1}^{l}\mu_n^l}.$$

Thus, R_n^l is obtained the normalized KL-Divergence as:

$$R_n^l = \frac{1}{2}\left[e^{-\mathrm{KL}_{(n,l)}^{\Lambda}} + e^{-\mathrm{KL}_{(n,l)}^{M}}\right], \tag{19}$$

where $\mathrm{KL}_{(n,l)}^{\Lambda}$ denotes the KL-Divergence between the learned and observed traffic packets regarding the packet inter-transmission times, and $\mathrm{KL}_{(n,l)}^{M}$ denotes that regarding the packet lengths. Thus, following [51] we have:

$$R_n^l = \frac{1}{2}\left[\frac{\Lambda_{(n,l)}^L}{\Lambda_{(n,l)}^O}\, e^{-\frac{\left(\Lambda_{(n,l)}^L - \Lambda_{(n,l)}^O\right)}{\Lambda_{(n,l)}^O}} + \frac{M_{(n,l)}^L}{M_{(n,l)}^O}\, e^{-\frac{\left(M_{(n,l)}^L - M_{(n,l)}^O\right)}{M_{(n,l)}^O}}\right] \tag{20}$$

4.2.2. Generalization ability of $IDAF-n$

The generalization ability G_n^l of the $IDAF-n$, evaluates how well it can generalize from the traffic it learns, so as to make accurate decisions. Thus, it is calculated as the average of the data adequacy Δ_n^l and the $IDAF-n$ expected performance κ_n^l:

$$G_n^l = \frac{\Delta_n^l + \kappa_n^l}{2}, \tag{21}$$

is Δ_n^l is the ratio of the total number of learnable parameters N_p to the total number of windows used for learning:

$$\Delta_n^l = \left[1 - \min\left(\frac{N_p}{|\mathcal{D}_n^l|}, 1\right)\right], \tag{22}$$

and κ_n^l is calculated from the historical learning error:

$$\kappa_n^l = 1 - \sum_{k=0}^{l}\left(\frac{1}{2}\right)^{(l-k+1)}\mathcal{E}(I_n^k, k) \tag{23}$$

4.3. $IDAF-n$'s Unsupervised Auto-Associative Local Learning Algorithm (LLA)

In every time window l, the parameters I_n^l of $IDAF-n$, for each node n, are obtained via the LLA, that updates the DRNN weights and SWBC parameters for window l using the local benign traffic $\mathcal{D}_n^l$ at node n.

To this end, for the connection weights of each hidden layer $h \in \{1,\ldots,H-1\}$, the following square cost with L1 regularization [49], is minimized using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [53]:

$$W_{(n,h)}^l = \underset{\{W:\, W\geq 0\}}{\operatorname{argmin}}\ \Big(\Big|\Big| [adj(\Psi(\hat{X}_{(n,h-1)}^l W_R)), \mathbf{1}_{|\mathcal{D}_n^l|}]\, W - \hat{X}_{(n,h-1)}^l \Big|\Big|_2^2 + \|W\|_1 \Big), \tag{24}$$

where

$$\hat{X}_{(n,h)}^l = [\hat{x}_{(n,h)}^k, \quad \forall k \in \mathcal{D}_n^l], \quad \forall h \geq 1, \tag{25}$$

$$\hat{X}_{(n,0)}^l = [x_n^k, \quad \forall k \in \mathcal{D}_n^l]. \tag{26}$$

and $\mathbf{1}_{|\mathcal{D}_n^l|}$ is a column vector of ones with length $|\mathcal{D}_n^l|$, and W_R is $(H \times H)$ matrix whose elements are randomly generated with uniform distribution in $[0, 1]$. Moreover, $adj(A)$ linearly maps the elements of matrix A to the range $[0, 1]$, then applies z-score and adds a positive constant to remove negativity.

After these connection weights are obtained, each resulting weight matrix $W_{(n,h)}^l$ is normalized for each hidden layer $h \in \{1, \ldots, H-1\}$:

$$W_{(n,h)}^l \leftarrow 0.1 \frac{W_{(n,h)}^l}{\max_{k \in \mathcal{D}_n^l} \left(\hat{X}_{(n,h)}^k\right)}, \tag{27}$$

and the connection weights of the output layer H are calculated with an extreme learning machine:

$$W_{(n,H)}^l = (\hat{X}_{(n,H-1)}^l)^+ X_n^l, \tag{28}$$

where A^+ is the pseudo-inverse of matrix A.

4.3.1. Computing the SWBC parameters

Auto-associative on-line sequential learning is the process that is used to train the DRNN. This process creates the learning dataset $\mathcal{D}_n^l$ composed of successive elements which are classified as benign by the DRNN, and then used for online learning to create the subsequent elements of the dataset. If the learning algorithm were "perfect", then this dataset would NOT contain any False Negatives, and we should select the decision threshold as being the largest value of the error term in the dataset created at the end of the lth training epoch $\mathcal{D}_n^l$ for the nth network node, i.e.:

$$\theta_n^l = Max_{\mathcal{D}_\mathbf{n}^\mathbf{l}} \{\zeta_n^k\}, \ where \ \zeta_n^k = |x_{n,i}^k - \hat{x}_{n,i}^k|. \tag{29}$$

However, we estimate that the DRNN is not "perfect", and hence that there may be some False Negative elements in each successively calculated dataset $\mathcal{D}_n^l$. Therefore, to be more sensitive to False Negatives, we select a smaller value of the threshold, that is given by:

$$\theta_n^l = Mean_{\mathcal{D}_\mathbf{n}^\mathbf{l}} \{\zeta_n^k\} + 2 \times Standard Dev_{\mathcal{D}_\mathbf{n}^\mathbf{l}} (\zeta_n^l). \tag{30}$$

Then, one also obtains the lower $Q_{n,i}^L$ and upper quartile $Q_{n,i}^U$ of $\{z_{n,i}^k\}_{k \in \mathcal{D}_n^l}$, so that the upper whisker $w_{n,i}^l$ is:

$$w_{n,i}^l = Q_{n,i}^U + \frac{3}{2}(Q_{n,i}^U - Q_{n,i}^L) \qquad \forall i \in \{1, 2, 3\} \tag{31}$$

As a last part of F^*, the parameters I_n^l of $IDAF-n$ at each window l, are updated using the parameters received from the federated nodes $\mathcal{M}_n^l$ through the following steps:

4.4. Data fusion with concurring nodes

DISFIDA's fusion rule F^* selects for each $IDAF-n$, the set of concurring nodes C_n^l, that achieve decisions similar to those of node n on the local traffic data of n, by evaluating the performance at each node $m \in \mathcal{M}_n^l$ on the local data at node n, over all time windows up to current window l:

$$C_n^l = \{m : \ \frac{1}{l} \sum_{k=1}^{l} \mathbf{1}\left(I_m^l(x_n^k) = y_n^k\right) \geq \Theta, \ \ \forall m \in \mathcal{M}_n^l\}, \tag{32}$$

where $I_m^l(x_n^k)$ is the decision of the IDS of node m for the input x_n^k.

Then, the rule F^* updates the parameters of I_n^l, separately for each segment by averaging it with the parameters of the closest node in the set C_n^l:

- For each layer $h \in \{1, \ldots, H-1\}$ of DRNN in I_n^l, the node m_h^* that has the closest connection weights $W_{(n,h)}^l$ with node n in window l is obtained:

$$m_h^* = \arg\min_{m \in C_n^l} \left(\left|\left| W_{(n,h)}^l - W_{(m,h)}^l \right|\right|_1 \right), \tag{33}$$

  and the connection weights $W_{(n,h)}^l$, are updated:

$$W_{(n,h)}^l \leftarrow c\, W_{(n,h)}^l + (1-c)\, W_{(m_h^*,h)}^l, \tag{34}$$

  where $0.5 \leq c \leq 1$ is a coefficient that prioritizes locally learned weights over federated weights.
- For each whisker $w_{n,i}^l$ of SWBC in I_n^l, the node m_i^* with the whisker value closest to $w_{n,i}^l$ is obtained:

$$m_i^* = \arg\min_{m \in C_n^l} \left(\left| w_{n,i}^l - w_{m,i}^l \right| \right), \tag{35}$$

  and $w_{n,i}^l$ is updated as

$$w_{n,i}^l \leftarrow c\, w_{n,i}^l + (1-c)\, w_{m_i^*,i}^l. \tag{36}$$

Table 5
Types and Properties of the Datasets.

Dataset	Attack	Duration secs	BenignPkts	AttackPkts
Kitsune	MIRAI	7137	121621	642516
Bot-IoT	DoSHTTP	2940	56	29706
Bot-IoT	DDoSHTTP	2520	55	19771
OpelAstra	DoS	414.65	787539	40016
Opel Astra Test	No Attack	414.65	806999	0
RenaultClio	DoS Attack	82.54	101926	40001
RenaultClio Test	No Attack	82.54	115971	0
Prototype Vehicle	DoS	82	25456	40037
Prototype Veh. Test	No Attack	82	28993	0

- For each whisker, the node m^*_θ that has the threshold value closest to the decision threshold θ^l_n is obtained:

$$m^*_\theta = \arg\min_{m \in C^l_n} \left(\left| \theta^l_n - \theta^l_m \right| \right), \tag{37}$$

and θ^l_n is updated:

$$\theta^l_n \leftarrow c\,\theta^l_n + (1-c)\,\theta^l_{m^*_\theta}. \tag{38}$$

In order adapt each $IDAF-n$ to the local *benign* network traffic of node n, the output layer weights of DRNN are updated with the ELM algorithm.

5. Experimental results

We now compare the performance of DISFIDA that uses the Fusion Rule F^*, against other benchmark methods. The evaluations use two relevant and distinct scenarios with DoS and DDoS attacks that target (1) an IoT network and (2) the CAN-bus of Smart Vehicles. In Table 5, we summarize the main characteristics of the datasets that are used in the evaluations.

The following Fusion Rules F are tested and compared:

- F_0 **The Non-Federated Approach** Contrary to DISFIDA and other federated approaches, F_0 is the conventional "non-fusion" approach, where neural weight federation does not take place since each intrusion detection system learns locally from its data, and does **not** fuse its own weights with those of other nodes.
- F_{AVG} **Average over All Collaborating Nodes** Called "Average" in the figures and tables, this approach updates the weights of $IDAF-n$ by averaging them with the neural weights of the other $N-1$ nodes: so that the weights at each layer h become:

$$W^l_{(n,h)} \leftarrow \frac{1}{N} \sum_{m \in \mathcal{N}} W^l_{(m,h)}, \tag{39}$$

Subsequently, the SWBC parameters are also updated in the same way:

$$w^l_{n,i} \leftarrow \frac{1}{N} \sum_{m \in \mathcal{N}} w^l_{m,i} \quad \forall i, \quad and \quad \theta^l_n \leftarrow \frac{1}{N} \sum_{m \in \mathcal{N}} \theta^l_n \tag{40}$$

After updating, the parameters for all the nodes are identical until the next learning window.
- F^+_{AVG} **Average with Closest Node (ACN)** This approach updates the weights at each $IDAF-n$ by taking the average with the closest parameters among all other $IDAF-v,\ v \neq m$. At time window l the node $m_l(n)$ is identified:

$$m_l(n) = \arg\min_{m \in \mathcal{N} \setminus n} \left(\sum_{h=1}^{H} \left|\left| W^l_{(n,h)} - W^l_{(m,h)} \right|\right|_1 + \sum_{i=1}^{3} \left| w^l_{n,i} - w^l_{m,i} \right| + \left| \theta^l_n - \theta^l_m \right| \right) \tag{41}$$

Then for each $n \in \mathcal{N}$ the neural weights if $IDAF-n$ are updated at time window l:

$$W^l_{(n,h)} \leftarrow \frac{W^l_{(n,h)} + W^l_{(m^*,h)}}{2}, \quad \forall h \tag{42}$$
$$w^l_{n,i} \leftarrow \frac{w^l_{n,i} + w^l_{m^*,i}}{2}, \quad \forall i \qquad \theta^l_n \leftarrow \frac{\theta^l_n + \theta^l_{m^*}}{2}$$

- F^L_{AVG} **Average with Closest Node per Layer (ACN-L)** updates each weight of $IDAF-n$ for each neuronal layer, and updates each whisker and threshold of SWBC, by individually taking the average with the parameter of the closest node with respect to that parameter. For each layer h of DRNN in I^l_n, the connection weights of this layer are updated using (34) for a value of m^*_h calculated using (33):

$$W^l_{(n,h)} \leftarrow \frac{W^l_{(n,h)} + W^l_{(m^*_h,h)}}{2}, \tag{43}$$

where

$$m^*_h = \arg \min_{m \in \mathcal{N} \setminus n} \left(\left|\left| W^l_{(n,h)} - W^l_{(m,h)} \right|\right|^2_2 \right). \tag{44}$$

Then, each whisker $w^l_{n,i}$ of SWBC in I^l_n is updated (36) for a value of m^*_i calculated using (35):

$$w^l_{n,i} \leftarrow \frac{w^l_{n,i} + w^l_{m^*_i,i}}{2}, \tag{45}$$

where

$$m^*_i = \arg \min_{m \in \mathcal{N} \setminus n} \left(\left|\left| w^l_{n,i} - w^l_{m,i} \right|\right|^2_2 \right). \tag{46}$$

The decision threshold θ^l_n is the updated by using (37) and (38):

$$\theta^l_n \leftarrow \frac{\theta^l_n + \theta^l_{m^*_\theta}}{2}, \quad where \quad m^*_\theta = \arg \min_{m \in \mathcal{N} \setminus n} \left(\left|\left| \theta^l_n - \theta^l_m \right|\right|^2_2 \right). \tag{47}$$

- F_S **Synchronous Federated:** This algorithm was introduced in [48]. We used it for comparison, because it achieves high intrusion detection performance, and performs a federated update in each time window with the algorithm detailed in [48], using the current parameters learned at all the nodes.
- F^* **DISFIDA** detailed in Section 4.

5.1. System configuration and parameter settings

DISFIDA uses a DRNN that was introduced for the first time, independently of any specific application, in [49] at each node n. The DRNN is structured in the present case as a 3 -level feedforward architecture composed of identical clusters of 10 neurons, whose parameters are set according to the results in [49]. The clusters are connected to other in a feedforward manner, while within each cluster the DRNN has soma-to-soma feedback between all neurons, represented by a triggering probability $p = 0.05$. The firing rate of each neuron inside the cluster is set to $r = 0.001$, and each neuron receives fixed external excitatory and inhibitory spike arrival rates set to $\lambda^+ = \lambda^- = 0.1$. At level h the neuronal clusters' output feeds into the clusters of the next level with the inhibitory weight matrix $W_{(n,h)}$ which is tuned with the DRNN LLA (Local Learning Algorithm that was also introduced in [49]. The DRNN was previously used for SYN attack detection, with the same parameters, and the same LLA, but with totally different learning and testing Datasets from the present work, in [50].

Furthermore, during the evaluation of DISFIDA, we set $K = 10$, the decision threshold to $\gamma = 0.5$, and the coefficient of weighted average to $c = 0.75$. This means that LLA only updates local parameters if at least half of the last 10 windows are classified as benign.

Our experiments were conducted on a computer equipped with 16 GB of RAM and an M1 Pro 8-core 3.2 GHz processor. In order to be as close to reality as possible, we incorporated threading that enables the concurrent operation of three nodes.

5.2. Datasets

To assess the viability of our developed concept, we employ two publicly accessible datasets, Kitsune [44,45] and Bot-IoT [46]. Our experiments involve three distinct attack datasets, namely Mirai, DoS HTTP, and DDoS HTTP. Each data corresponds to an individual node within the DISFIDA architecture. In essence, we examine the collaboration of three nodes, each representing an IoT network with data sourced from a public dataset.

In the DISFIDA architecture, we have considered three nodes, each of which receives attacks that are emulated by distinct publicly available data sets. The first node is attacked by the "Mirai Botnet", sourced from the Kitsune dataset [45] comprised of 764,137 packets transmitted by 107 unique IP addresses over a duration of 7137 seconds, i.e. around 2 hours. For the second and third nodes within this experiment with DISFIDA, we employ "DoS HTTP" and "DDoS HTTP" attacks from the Bot-IoT dataset [46]. The resulting collaborative system using DISFIDA is represented in Fig. 3. The DoS HTTP attack data consists of 29,762 packets transmitted over 49 minutes, while the DDoS HTTP attack dataset has 19,826 packets transmitted over 42 minutes. Each of these attack datasets initially start with malicious traffic, while the learning algorithm used by DISFIDA learns with benign traffic. Thus, we need a short cold-start, so that we have reversed each dataset along the time axis resulting in an initial short sequence of benign packets, before each attack actually starts.

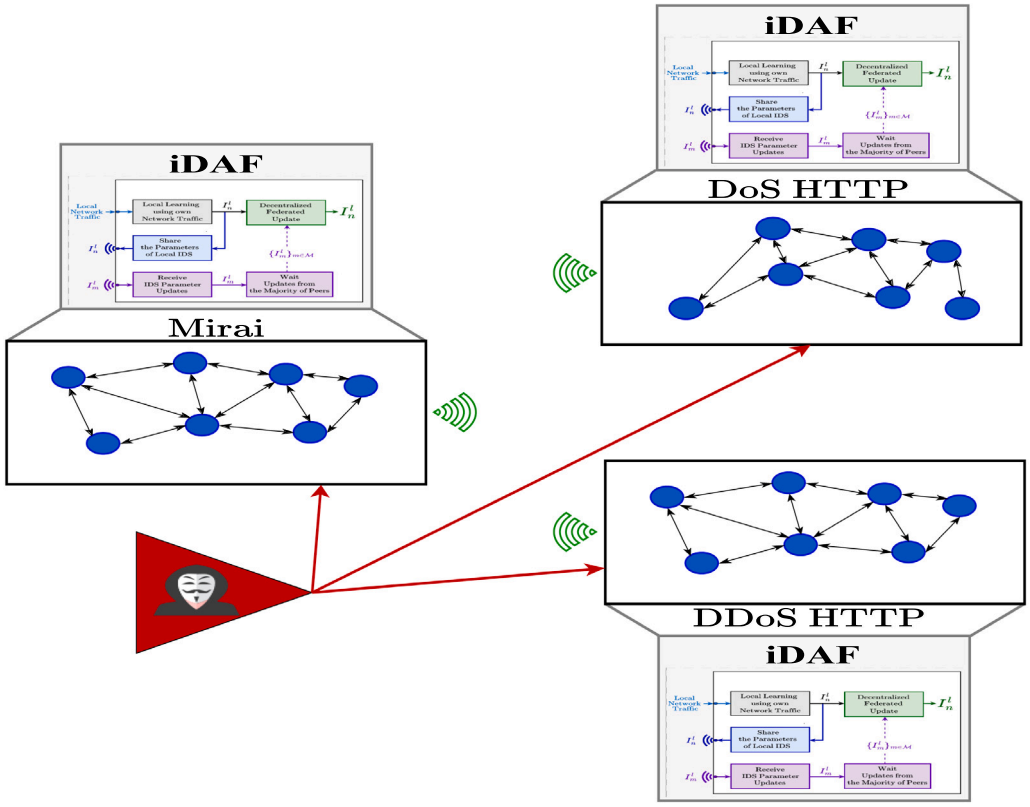


Fig. 3. An IoT network with three nodes, where $IDAF-1$, $IDAF-2$ and $IDAF-3$ are targeted by three distinct attacks: respectively the Mirai, DoS HTTP and DDoS HTTP attacks.

The value of T_n in seconds is set pragmatically based on two considerations: (1) the number of packets in the window must be sufficiently large for reliable decision making, and this depends on the traffic rate in the given dataset, (2) in addition to having enough packet arrivals, the related computations must fit into the time window for all the N nodes. Based on these considerations we chose T_n equal to 23 secs for the experiments with the Mirai dataset, 9 secs for the DoS HTTP dataset, and 8 secs for the DDoS HTTP dataset. It is important to note that both datasets provide a binary ground truth, denoted as $a(p_n^t)$, for each packet p_n^t, which is determined by the dataset providers, indicates whether the packet is classified as a normal "benign" packet or "malicious", signifying an ongoing attack.

5.3. Connected smart vehicles for patient transport

Calling and dispatching smart vehicles to transport patients in a timely fashion, will be a part of the health systems in the future. Thus, in addition to examining IoT datasets, we also evaluate our DISFIDA system to detect CAN-bus DoS attacks, provided in the dataset [47]. This dataset, released by researchers from Eindhoven University of Technology, comprises data collected from two vehicles, Opel Astra and Renault Clio, and a CAN-bus prototype they developed.

For the performance evaluation, we use the "testing.log" and "dosattack.log" files from Opel Astra, Renault, and the Prototype Vehicles. For each car, the testing.log file is comprised of only normal CAN-bus data while the dosattack.log file contains both normal and attack data, which are clearly labeled. Accordingly, we consider each file as an individual car as represented in Fig. 4.

5.4. Comparison of DISFIDA with other weight fusion methods for IoT

We now evaluate the intrusion detection performance of the proposed DISFIDA system for an IoT network using the datasets discussed in Section 5.2. We recall that after the learning is complete at each node and each node transmits its weights to all the other ones, then DISFIDA assigns the rule F^*, that fuses the node weights at each node n with the weights of other nodes, based on the greatest similarities between weights.

In Fig. 5, we present the performance of DISFIDA compared with benchmark methods with respect to True Positive Rate (TPR) and the True Negative Rate (TNR):

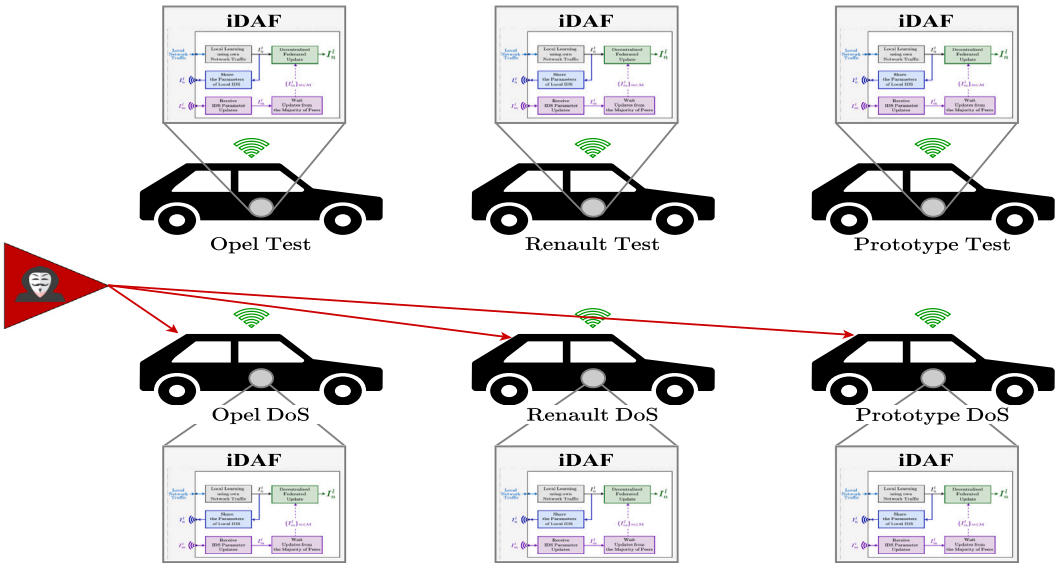


Fig. 4. Schematic description of six connected Smart Vehicles, which constitute the $IDAF-n$ nodes for $1 \leq n$ 6, targeted by distinct CAN-bus DoS attacks Opel, Renault and Prototype.

Table 6
Comparison of the best-performing methods with respect to Precision (Prec), Recall (Rec), F1 Score, and MCC metrics for the case of Collaborating IoT Networks. In this Table, Synch refers to Synchronous Federated Learning, while No Fed refers to the case where Federated Learning is not used.

Methods	Mirai				DoS HTTP				DDoS HTTP			
	Prec	Rec	**F1**	MCC	Prec	Rec	**F1**	MCC	Prec	Rec	**F1**	MCC
DISFIDA	0.99	1	0.99	0.99	0.91	1	0.95	0.93	0.69	0.90	0.78	0.63
Synchronous	0.943	1	0.97	0.96	0.84	0.99	0.91	0.86	0.80	0.92	0.86	0.76
No Fed	0.94	1	0.97	0.96	0.97	0.39	0.55	0.534	0.60	0.26	0.37	0.2

- The results in the top figure displaying the TPR, show that DISFIDA's intrusion detection algorithm IDAF, and other federated learning methods, detect malicious traffic with very high accuracy.
- F_0 "No-Federated" approach results in unacceptably low TPR.
- The bottom figure shows that DISFIDA, the Synchronous and 'No Federated" approach, yield high TNR (low false alarms).
- The standard averaging approach for federated updates has unacceptably high false alarms, and therefore is not suitable for combining model parameters learned from different network traffic data at distinct nodes.

In summary, these results highlight that DISFIDA and Synchronous federated learning [48] are very promising learning methods for federated attack detection.

Table 6 displays a detailed comparison of the three best-performing methods, namely DISFIDA, Synchronous, and No Federated, based on Precision, Recall, F1 Score, and MCC metrics. The results in this table show that the federated learning IDSs, namely DISFIDA and Synchronous, are significantly superior to the Non-federated method. For Mirai and DoS HTTP networks, DISFIDA makes more accurate decisions with higher Precision and Recall compared to the Synchronous method. Also, the very stable performance of DISFIDA revealed by F1 Score and MCC metrics for these network shows that it provides accurate attack detection with a low false alarm rate. On the other hand, for DDoS HTTP, DISFIDA underperforms the Synchronous method, especially in terms of the precision of its decisions, which leads to a high number of false alarms.

The performance evaluation results for collaborating IoT network show that DISFIDA successfully detects both DoS and DDoS (including Botnet) attacks with high accuracy and acceptably low false alarm rates. In addition, DISFIDA provides highly competitive (often better) performance compared to state-of-the-art Synchronous federated learning IDS. Since most standalone network operate asynchronously, DISFIDA shall be the main federated learning IDS for distributed collaborative IoT networks.

5.5. Performance evaluation for connected smart vehicles

We will now test the performance of DISFIDA for detecting CAN-bus attacks targeting connected Vehicles, which may be used to transport patients infuture integrated health systems. Using datasets given in Section 5.3 are used, we first present the evaluation

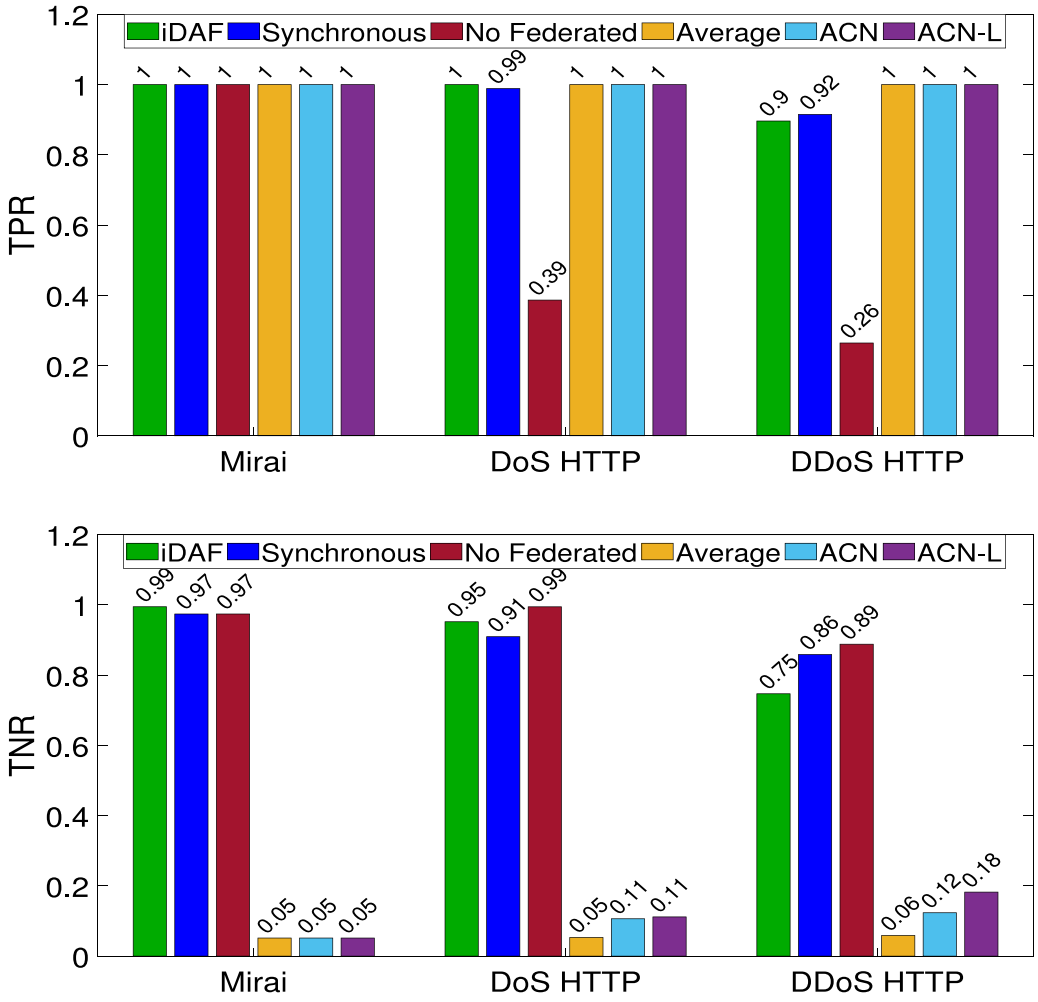


Fig. 5. Performance comparison of DISFIDA with the benchmark methods (Synchronous, No Federated, Average, ACN, and ACN-L) with respect to TPR (top) and TNR (bottom) for Mirai, DoS HTTP and DDoS HTTP data.

results on three experimental data involving DoS attacks in Fig. 6, where DISFIDA is tested for a system of six connected Vehicles that would be operating within the same IoV network.

For critical infrastructures such as the CAN-bus of smart Vehicles, misclassification of malicious activity (one minus the TPR), can be more dangerous than raising false alarms.

Thus to examine this aspect further, DISFIDA and the different detection algorithms are compared, for several public datasets, based on the TPR and TNR in Fig. 6, showing characteristics similar to those for collaborating IoT networks given in Fig. 5. At the top, we see that DISFIDA, Synchronous, and No-Federated methods have intrusion detection performance, while at the bottom we see that the three averaging-based methods result in high false alarm rate, and that DISFIDA has a slightly higher false alarm rate for Renault and Prototype Vehicles, while it detects malicious traffic 100% accurately (top) for all three classes of vehicles. Moreover (bottom), for intrusion detection from the Opel DoS dataset, the DISFIDA, Synchronous and No-Federated approaches successfully detect malicious traffic with almost no false alarms. On the other hand (top), DISFIDA detects malicious traffic very accurately, but with a high false alarm rate (bottom) for both Renault and Prototype Vehicles. Meanwhile (top), the No-Federated and Synchronous methods have unacceptably poor detection performance, achieving TPRs of about 0.1 for the Renault and Prototype cases.

Furthermore, Fig. 7 presents the performance of all methods compared for the Vehicles that are not attacked within the data, called Opel Benign, Renault Benign, and Prototype Benign. The results in this figure show that although DISFIDA has a higher false alarm rate for Renault, it makes almost perfect decisions with low false alarms for both Opel and Prototype Vehicles.

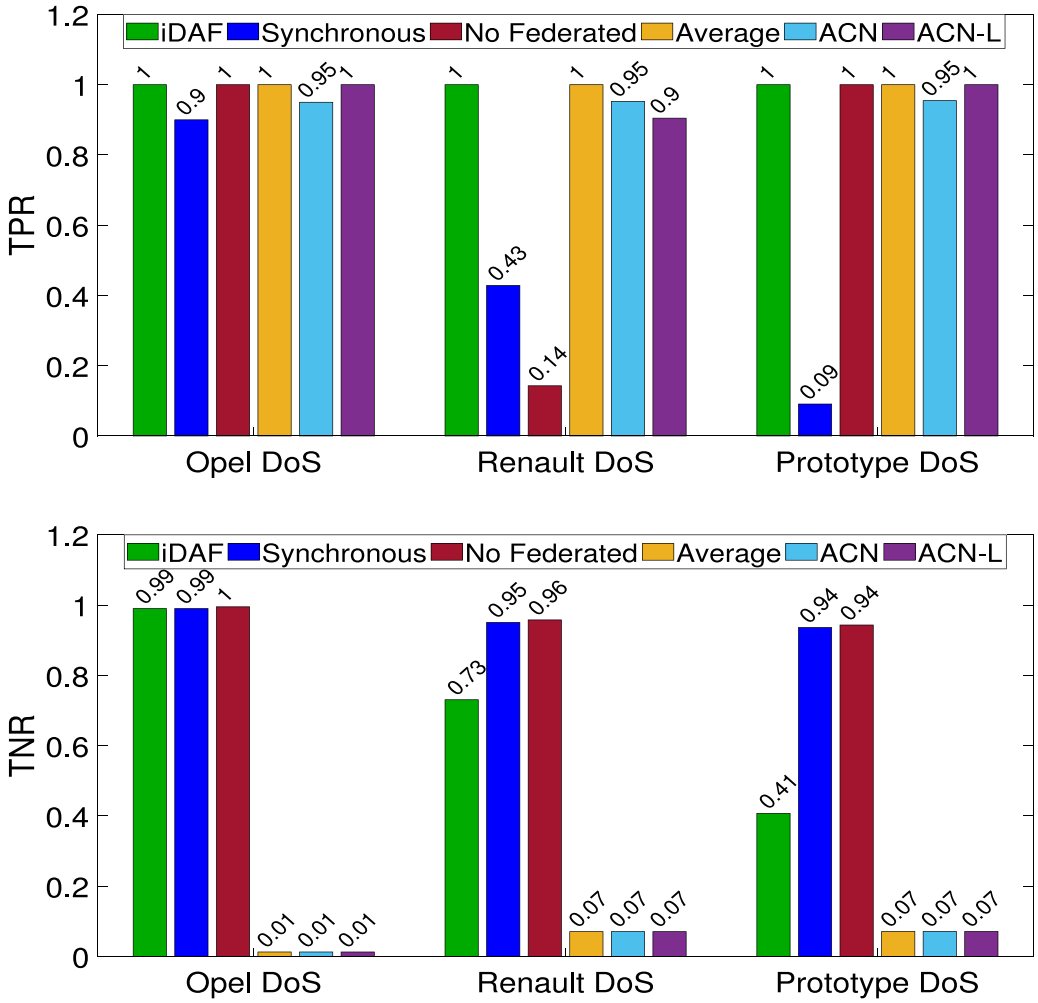


Fig. 6. Performance comparison of DISFIDA with benchmark methods (Synchronous, No Federated, Average, ACN, and ACN-L) with respect to TPR (top) and TNR (bottom) for Opel DoS, Renault DoS, and Prototype DoS data.

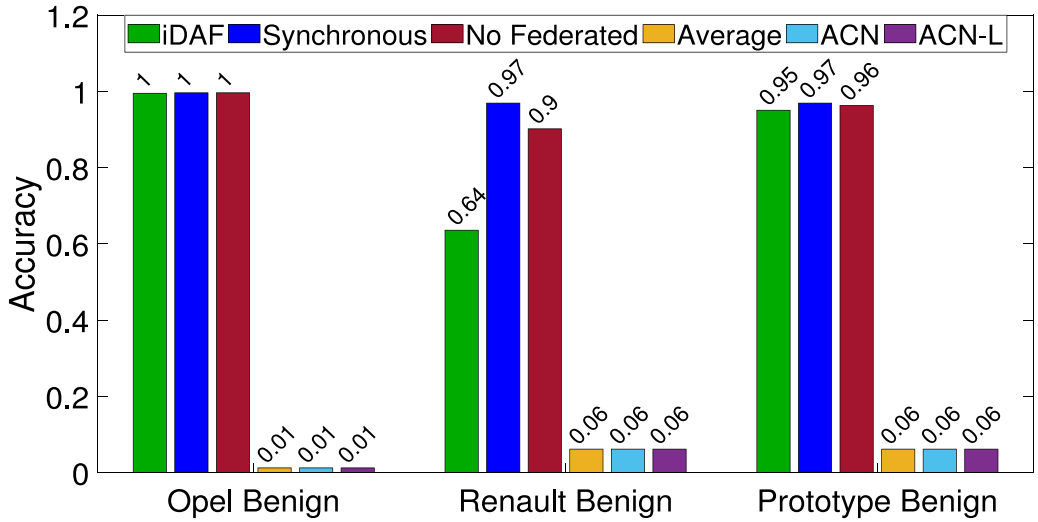


Fig. 7. Performance comparison of DISFIDA with benchmark methods (Synchronous, No Federated, Average, ACN, and ACN-L) with respect to TNR (top) and TPR (bottom) for Opel Benign, Renault Benign, and Prototype Benign data.

Table 7
Total traffic, in kilobytes (KB), transmitted by the six Connected Smart Vehicles to exchange non-encrypted up-to-date IDAF parameters.

	Opel		Renault		Prototype	
	DoS	Benign	DoS	Benign	DoS	Benign
DISFIDA	4.25	4.01	1.18	1.18	8.73	18.17
Synchronous	189.04	194.46	34.69	37.29	35.87	37.29
Average	2.36	2.36	2.36	2.36	2.36	2.36
ACN	2.60	2.36	2.60	2.36	2.60	2.36
ACN-L	2.36	2.36	2.83	2.36	2.36	2.36

5.6. DISFIDA's communication and computation overhead

We also analyze the communication overhead caused by using DISFIDA or other federated learning methods. To this end, we assume that a privacy-preserving scheme has been implemented prior to the transmission of model updates in order to protect the outcomes of the training process.

Fully Homomorphic Encryption (FHE) can be used in this context to achieve robust security and confidentiality, by allowing arithmetic computations on encrypted data without the need for decryption [54], so that results match those obtained from the corresponding unencrypted data. However, FHE scheme requires substantial computation, and, results in exponential growth in the size of the transmitted message. In the present work, the TenSEAL library [55], which relies on CKKS-Scheme [56], for the implementation of FHE in Python, has been used.

During the performance evaluations, at each parameter update, the list of all 49 parameters, each presented as a floating point number, is first encrypted using FHE. Note that while the unencrypted parameters can be represented with 196 bytes using the Struct library [57], the parameters encrypted using FHE are represented with 334.5 kilobytes (KB).

Since the Transmission Control Protocol (TCP)/Internet Protocol (IP) is used for communication between collaborating nodes, for each federated update we create a TCP/IP packet using the Scapy library [58], that contains the IDAF parameters. As a result, each assembled update, including the 40-byte TCP/IP header, requires the transmission of 334.54 KB bytes when using FHE or 236 bytes when not using FHE. It should be noted that although it is not necessary to use encryption as federated learning already ensures the confidentiality of local data by exchanging only the model parameters, the use of FHE or another encryption method is needed to obtain an entirely secure federated learning system.

Fig. 8 presents the percentage communication overhead for DISFIDA and other methods without encryption or with FHE for collaborating IoT networks. For each IoT network, the communication overhead is measured as the ratio of the total amount of traffic used for transmitting and receiving federated updates to the size of all normal (benign) traffic, including the traffic of federated updates.

As the results in Fig. 8 show, the proposed DISFIDA system requires the communication of the lowest amount of traffic compared to other federated learning methods. In Fig. 8 (top), when local weights are not encrypted before transmission, the communication overhead of DISFIDA is below 0.03% for all IoT networks. When local weights are encrypted using FHE before transmission, the results in Fig. 8 (bottom) show that all methods have higher overhead now due to the large packet size required by the encryption. For Mirai, one can be seen that unified update communication of any method other than DISFIDA dominates normal traffic. It is also seen that the communication load of the Synchronous method is significantly high for all networks.

With or without encryption, we see that the proposed DISFIDA system uses much less traffic than other methods. Also, Synchronous, the most competitive method in terms of detection performance, has an order of magnitude more overhead than DISFIDA. The low communication overhead of DISFIDA is mainly based on the use of self-supervised learning for local parameter updates. Since this learning approach, based on the SSID framework [51], eliminates unnecessary parameter updates, it has a positive impact on both the detection performance, and the use of the communication medium.

Furthermore, Table 7 and Table 8 display the total size of data packets transmitted by each smart car and carrying federated parameter updates without encryption and with FHE, respectively. The results in Table 7 shows that the connected Vehicles using DISFIDA transmit much less traffic compared to those using Synchronous method. The similar conclusion is also obtained from Table 8. On the other hand, note that the total amount of traffic used by each method is three orders of magnitude higher when FHE is utilized in Table 8.

Moreover, although the Average, ACN, and ACN-L methods require the least amount of communication for federated learning in regarding connected Smart Vehicles, in Section 5.5 we see that their intrusion detection performance is excessively low.

5.7. Computation time of federated learning

Finally, we briefly analyze the computation time required by each method for a single federated update. To this end, we present in Table 9 the average of the computation times taken for federated updates (executed via DAFU) over all collaborating nodes in the system.

Measurements in Table 9 show that each federated parameter update in the proposed DISFIDA system takes approximately 3.6 ms for collaborative IoT network and 32 ms for connected smart Vehicles. All methods, including DISFIDA, spend more time on federated updates in connected smart Vehicles than in collaborative IoT networks, as expected, due to the larger number of collaborating nodes. Moreover, although the calculation time of DISFIDA is higher than that of the Average, ACN and ACN-L methods, it is an order of magnitude less than the computation time of the Synchronous method.

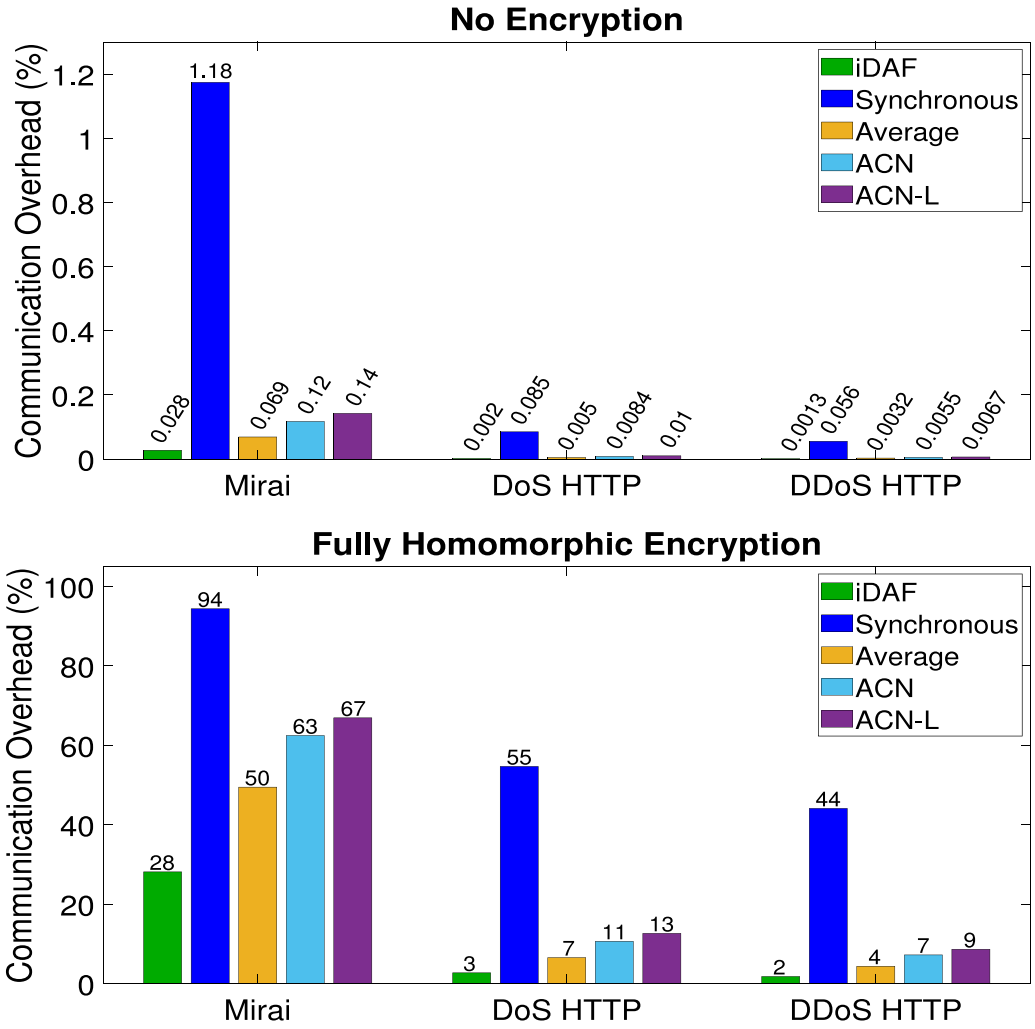


Fig. 8. Percentage ratio of traffic used for forwarding the federated updates to other nodes using (a) NOR encryption and (b) Full Homomorphic Encryption (FHE).

Table 8
Total amount of traffic, in megabytes (MB), transmitted by Connected Smart Vehicles to exchange FHE encrypted up-to-date IDAF parameters.

	Opel		Renault		Prototype	
	DoS	Benign	DoS	Benign	DoS	Benign
DISFIDA	6.02	5.69	1.67	1.67	12.38	25.76
Synchronous	267.97	275.66	49.18	52.86	50.85	52.86
Average	3.35	3.35	3.35	3.35	3.35	3.35
ACN	3.68	3.35	3.68	3.35	3.68	3.35
ACN-L	3.35	3.35	4.02	3.35	3.35	3.35

6. Conclusions and future work

The proposed novel DISFIDA system described in this paper, is a decentralized and asynchronously trained frontline self-supervised federated learning system. It allows distributed networked nodes to collaboratively learn intrusion detection and increase their overall security while preserving the confidentiality of each node's local data.

Such features are important to all medical applications including medical device IoT networks, networks for medical data, and also smart vehicles that may be used to convey patients for medical diagnostics or treatment.

Each participating node $IDAF - n$ in DISFIDA will first learn its local incoming network traffic to create an auto-associative memory regarding normal (benign) traffic, using a self-supervised auto-associative DRNN [49] with no human intervention. Then,

Table 9
Average time spent, in milliseconds (ms), by each method for the federated update in a single window.

	Collaborating IoT network	Connected smart vehicles
DISFIDA	3.56 ms	32.1 ms
Synchronous	29.7 ms	261.21 ms
Average	0.0335 ms	0.057.4 ms
ACN	0.0691 ms	0.2688 ms
ACN-L	0.107 ms	0.2998 ms

the internal parameters that are learned, are shared with other nodes. Since all nodes in this system work and learn asynchronously, each node that completes its local learning shares the locally learned internal parameters with other nodes. A local update on every local IDAF parameter, is then carried out, when a given node receives updates from the *majority* of its collaborating nodes.

DISFIDA was evaluated for DoS and DDoS attacks for two attack scenarios: (1) The collaboration of three device IoT Networks in the presence of Kitsune attacks [44,45], and the Bot-IoT [46] attacks, and for (2) Six connected Smart Vehicles for conveying patients, based on the CAN-bus attacks [47].

In both scenarios, we measured DISFIDA's intrusion detection performance, as well as the resulting computation and communication overhead. We also compared its performance against five benchmark methods, including the state-of-the-art decentralized synchronous federated learning approach, leading to the following conclusions:

- DISFIDA achieves high intrusion detection accuracy by perfectly detecting malicious traffic for almost all nodes with an acceptably low false alarm rate. It appears that the DISFIDA approach of choosing to update the local parameters with those of other nodes having similar weights, provides higher resulting accuracy than the case where the local and incoming (neighbors') weights are averaged.
- DISFIDA is efficient with under 0.03% communication overhead when additional data encryption is not used.
- In addition, for a system where a IoT network includes devices and smart vehicles, it carries out joint federated updates in a time ranging between 3.56 ms and 32.1 ms.
- It eliminates the need for large local training data for individual nodes while preventing data leaks and confidentiality breaches, that are crucial in connected health informatics and devices.
- DISFIDA stands out among other federated learning methods for cybersecurity since it does not require human intervention or node synchronization, and delivers outstanding performance.

Therefore, DISFIDA can be recommended as a key enabler in the development of fully online and collaborative learning Artificial Intelligence (AI)-based security for IoT and IoV networks in secure privacy preserving health systems.

In future work, in order to mitigate the devastating effects of DoS attacks on victim devices [59,60], the intrusion decisions of DISFIDA can be combined with a mitigation algorithm to create a collaborative learning system for optimum Intrusion Detection and Prevention System.

CRediT authorship contribution statement

Erol Gelenbe: Writing – review & editing, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Baran Can Gül:** Methodology, Investigation, Formal analysis. **Mert Nakıp:** Writing – original draft, Validation, Software, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M.L. Machal, An overview about connected medical devices and their risks, in: Studies in Health Technology Informatics, in: Healthcare Transformation with Informatics and Artificial Intelligence, vol. 305, 2023, pp. 119–122, http://dx.doi.org/10.3233/SHTI230438.
[2] Global medical device market to grow 4.5% a year, 2018, URL https://www.24x7mag.com/medical-equipment/.
[3] NHS England, CyberSecurity, 2023, URL https://www.england.nhs.uk/long-read/cyber-security/.
[4] BBC News, Dick Cheney: Heart Implant Attack was Credible, 2013, URL https://www.bbc.co.uk/news/technology-24608435.
[5] A. Razaque, F. Amsaad, M. Khan, S. Hariri, S. Chen, C. Siting, X. Ji, Survey: Cybersecurity vulnerabilities, attacks and solutions in the medical domain, IEEE Access 7 (2019) 168774–168797.

[6] A.I. Newaz, A.K. Sikder, M.A. Rahman, A.S. Uluagac, A survey on security and privacy issues in modern healthcare systems: Attacks and defenses, 2020, arXiv:2005.07359.

[7] ExtraHop, Denial of Service Attack: Definition, Examples, and Prevention, 2022, [Online]. Available: https://www.extrahop.com/resources/attacks/dos/. (Accessed: 09 March 2023),

[8] O. Yoachimik, DDoS attack trends for 2022 Q2, 2022, [Online]. Available: https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/. (Accessed: 09 March 2023),

[9] M.A. Alsoufi, S. Razak, M.M. Siraj, I. Nafea, F.A. Ghaleb, F. Saeed, M. Nasser, Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review, Appl. Sci. 11 (18) (2021) 8383.

[10] P. Maniriho, E. Niyigaba, Z. Bizimana, V. Twiringiyimana, L.J. Mahoro, T. Ahmad, Anomaly-based intrusion detection approach for iot networks using machine learning, in: 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia, CENIM, IEEE, 2020, pp. 303–308.

[11] E. Gelenbe, M. Nakıp, Traffic based sequential learning during botnet attacks to identify compromised IoT devices, IEEE Access 10 (2022) 126536–126549.

[12] A. Nisioti, A. Mylonas, P.D. Yoo, V. Katos, From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods, IEEE Commun. Surv. Tutor. 20 (4) (2018) 3369–3388.

[13] I.H. Sarker, CyberLearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks, Internet Things 14 (2021) 100393.

[14] E. Gelenbe, M. Nakıp, G-networks can detect different types of cyberattacks, in: 2022 30th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, IEEE, 2022, pp. 9–16.

[15] P. Natsvias, et al., Comprehensive user requirements engineering methodology for secure and interoperable health data exchange, BMC Med. Inform. Decis. Mak. 18 (1) (2018) 85, http://dx.doi.org/10.1186/s12911-018-0664-0.

[16] M. Nalin, et al., The European cross-border health data exchange roadmap: Case study in the Italian setting, J. Biomed. Inform. (ISSN: 1532-0464) 94 (2019) 103183, http://dx.doi.org/10.1016/j.jbi.2019.103183.

[17] D.C. Nguyen, et al., Federated learning for smart healthcare: A survey, ACM Comput. Surv. (ISSN: 0360-0300) 55 (3) (2022) http://dx.doi.org/10.1145/3501296.

[18] A. Frötscher, B. Monschiebl, A. Drosou, E. Gelenbe, M.J. Reed, M. Al-Naday, Improve cybersecurity of c-its road side infrastructure installations: the serIoT-secure and safe IoT approach, in: 2019 IEEE International Conference on Connected Vehicles and Expo, ICCVE, IEEE, 2019, pp. 1–5.

[19] S.R. Pokhrel, J. Choi, Federated learning with blockchain for autonomous vehicles: Analysis and design challenges, EEE Trans. Commun. 68 (8) (2020) 4734–4746, http://dx.doi.org/10.1109/TCOMM.2020.2990686.S2CID219006840.

[20] S. Savazzi, M. Nicoli, V. Rampa, Federated learning with cooperating devices: A consensus approach for massive IoT networks, IEEE Internet Things J. (5) (2020) 4641–4654.

[21] Z. Xu, F. Yu, J. Xiong, X. Chen, Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration, in: 2021 58th ACM/IEEE Design Automation Conference, DAC, 2021, pp. 997–1002, http://dx.doi.org/10.1109/DAC18074.2021.9586241.

[22] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends® Mach. Learn. 14 (1–2) (2021) 1–210.

[23] E.T. Martínez Beltrán, M.Q. Pérez, P.M.S. Sánchez, S.L. Bernal, G. Bovet, M.G. Pérez, G.M. Pérez, A.H. Celdrán, Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges, IEEE Commun. Surv. Tutor. 25 (4) (2023) 2983–3013, http://dx.doi.org/10.1109/COMST.2023.3315746.

[24] S. Safavat, D.B. Rawat, Asynchronous federated learning for intrusion detection in vehicular cyber-physical systems, in: IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2023, pp. 1–6, http://dx.doi.org/10.1109/INFOCOMWKSHPS57453.2023.10225917.

[25] S.S. Nivaashini M, FEDDBN-IDS: Federated deep belief network-based wireless network intrusion detection system, Secur. Saf. 1 (2023) http://dx.doi.org/10.21203/rs.3.rs-3136628/v1.

[26] S. Hajj, J. Azar, J. Bou Abdo, J. Demerjian, C. Guyeux, A. Makhoul, D. Ginhac, Cross-layer federated learning for lightweight iot intrusion detection systems, Sensors 23 (16) (2023) 7038.

[27] M.A. Merzouk, F. Cuppens, N. Boulahia-Cuppens, R. Yaich, Parameterizing poisoning attacks in federated learning-based intrusion detection, in: Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES '23, Association for Computing Machinery, New York, NY, USA, ISBN: 9798400707728, 2023, http://dx.doi.org/10.1145/3600160.3605090.

[28] R. Taheri, M. Shojafar, M. Alazab, R. Tafazolli, FED-IIoT: A robust federated malware detection architecture in industrial IoT, IEEE Trans. Ind. Inform. 17 (12) (2020) 8442–8452.

[29] Z. Li, X. Wu, C. Jiang, Efficient poisoning attacks and defenses for unlabeled data in ddos prediction of intelligent transportation systems, Secur. Saf. 1 (2022) 2022003, http://dx.doi.org/10.1051/sands/2022003.

[30] E.M. Campos, P.F. Saura, A. González-Vidal, J.L. Hernández-Ramos, J.B. Bernabé, G. Baldini, A. Skarmeta, Evaluating federated learning for intrusion detection in internet of things: Review and challenges, Comput. Netw. 203 (2022) 108661.

[31] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, DÏoT: A federated self-learning anomaly detection system for IoT, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2019, pp. 756–767.

[32] V. Mothukuri, P. Khare, R.M. Parizi, S. Pouriyeh, A. Dehghantanha, G. Srivastava, Federated-learning-based anomaly detection for iot security attacks, IEEE Internet Things J. 9 (4) (2021) 2545–2554.

[33] J. Li, L. Lyu, X. Liu, X. Zhang, X. Lyu, FLEAM: A federated learning empowered architecture to mitigate DDoS in industrial IoT, IEEE Trans. Ind. Inform. 18 (6) (2021) 4059–4068.

[34] G. Lu, Z. Xiong, R. Li, N. Mohammad, Y. Li, W. Li, DEFEAT: A decentralized federated learning against gradient attacks, High-Confidence Comput. (2023) 100128.

[35] Z. Lian, C. Su, Decentralized federated learning for internet of things anomaly detection, in: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, 2022, pp. 1249–1251.

[36] R. Al Mallah, D. López, Blockchain-based monitoring for poison attack detection in decentralized federated learning, in: 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME, IEEE, 2022, pp. 1–6.

[37] M.U. Nasir, S. Mehmood, M.A. Khan, M. Zubair, F. Khan, Y. Lee, Network intrusion detection empowered with federated machine learning, 2023, http://dx.doi.org/10.21203/rs.3.rs-3350992/v1.

[38] M. Abdel-Basset, N. Moustafa, H. Hawash, I. Razzak, K.M. Sallam, O.M. Elkomy, Federated intrusion detection in blockchain-based smart transportation systems, IEEE Trans. Intell. Transp. Syst. 23 (3) (2022) 2523–2537, http://dx.doi.org/10.1109/TITS.2021.3119968.

[39] M. Al-Hawawreh, M.S. Hossain, Federated learning-assisted distributed intrusion detection using mesh satellite nets for autonomous vehicle protection, IEEE Trans. Consum. Electron. (2023) 1, http://dx.doi.org/10.1109/TCE.2023.3318727.

[40] F.A.M. Do Nascimento, F. Hessel, Decentralized federated learning for intrusion detection in IoT-based systems: A review, in: 2022 IEEE 8th World Forum on Internet of Things, WF-IoT, 2022, pp. 1–6, http://dx.doi.org/10.1109/WF-IoT54382.2022.10152174.

[41] W. Yamany, M. Keshk, N. Moustafa, B. Turnbull, Swarm optimisation-based federated learning for the cyber resilience of internet of things systems against adversarial attacks, IEEE Trans. Consum. Electron. (2023) 1, http://dx.doi.org/10.1109/TCE.2023.3319039.

[42] Q. Zhang, Y. Wang, T. Wei, J. Wen, J. Chen, X. Qiu, IoT intrusion detection based on personalized federated learning, in: 2023 24st Asia-Pacific Network Operations and Management Symposium, APNOMS, IEEE, 2023, pp. 326–329.
[43] B. Li, W. Gao, J. Xie, M. Gong, L. Wang, H. Li, Prototype-based decentralized federated learning for the heterogeneous time-varying IoT systems, IEEE Internet Things J. (2023) 1, http://dx.doi.org/10.1109/JIOT.2023.3313118.
[44] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An ensemble of autoencoders for online network intrusion detection, in: The Network and Distributed System Security Symposium (NDSS) 2018, 2018.
[45] Kitsune Network Attack Dataset, 2020, [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune. (Accessed: 08 December 2023),
[46] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset, Future Gener. Comput. Syst. (ISSN: 0167-739X) 100 (2019) 779–796, http://dx.doi.org/10.1016/j.future.2019.05.041.
[47] G. Dupont, A. Lekidis, J.J. den Hartog, S.S. Etalle, Automotive Controller Area Network (CAN) Bus Intrusion Dataset v2, 2019, http://dx.doi.org/10.4121/uuid:b74b4928-c377-4585-9432-2004dfa20a5d.
[48] M. Nakip, B. Gul, E. Gelenbe, Decentralized online federated G-network learning for lightweight intrusion detection, in: 2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 1–8, http://dx.doi.org/10.1109/MASCOTS59514.2023.10387644, URL https://arxiv.org/pdf/2306.13029.
[49] E. Gelenbe, Y. Yin, Deep learning with random neural networks, in: 2016 International Joint Conference on Neural Networks, IJCNN, 2016, pp. 1633–1638, http://dx.doi.org/10.1109/IJCNN.2016.7727393.
[50] O. Brun, Y. Yin, E. Gelenbe, Deep learning with dense random neural network for detecting attacks against IoT-connected home environments, Procedia Comput. Sci. 134 (2018) 458–463.
[51] M. Nakıp, E. Gelenbe, Online self-supervised learning in machine learning intrusion detection for the internet of things, 2023, arXiv preprint arXiv:2306.13030.
[52] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[53] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202.
[54] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, STOC '09, Association for Computing Machinery, New York, NY, USA, ISBN: 9781605585062, 2009, pp. 169–178, http://dx.doi.org/10.1145/1536414.1536440.
[55] A. Benaissa, B. Retiat, B. Cebere, A.E. Belfedhal, TenSEAL: A library for encrypted tensor operations using homomorphic encryption, 2021, arXiv:2104.03152.
[56] J.H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic encryption for arithmetic of approximate numbers, in: T. Takagi, T. Peyrin (Eds.), Advances in Cryptology – ASIACRYPT 2017, Springer International Publishing, Cham, 2017, pp. 409–437.
[57] struct — Interpret bytes as packed binary data, 2023, [Online]. Available: https://docs.python.org/3/library/struct.html. (Accessed: 18 December 2023),
[58] Scapy, 2023, [Online]. Available: https://github.com/secdev/scapy. (Accessed: 18 December 2023),
[59] M. Nasereddin, M. Nakıp, E. Gelenbe, Measurement based evaluation and mitigation of flood attacks on a lan test-bed, in: 2023 IEEE 48th Conference on Local Computer Networks, LCN, IEEE, 2023, pp. 1–4.
[60] E. Gelenbe, M. Nasereddin, Protecting IoT servers against flood attacks with the quasi deterministic transmission policy (best paper award, IEEE trustcom 2023), in: The 22nd IEEE International Conference on Trust, Security and Privacy Computing and Communications, Vol. 2023, TrustCom-2023, November 2023, Exeter, UK, 2024, pp. 1–8, https://arxiv.org/pdf/2306.11007.pdf.