Blockchain and Deep Learning for Secure Communication in Digital Twin Empowered Industrial IoT Network

Article *in* IEEE Transactions on Network Science and Engineering · January 2022

DOI: 10.1109/TNSE.2022.3191601

CITATIONS
0

READS
20

6 authors, including:

Prabhat Kumar
LUT university
27 PUBLICATIONS **475** CITATIONS
SEE PROFILE

Randhir Kumar
National Institute of Technology Raipur
37 PUBLICATIONS **692** CITATIONS
SEE PROFILE

Sahil Garg
École de Technologie Supérieure
165 PUBLICATIONS **4,084** CITATIONS
SEE PROFILE

Blockchain and Deep Learning for Secure Communication in Digital Twin Empowered Industrial IoT Network

Prabhat Kumar, Randhir Kumar, Abhinav Kumar, A. Antony Franklin, Sahil Garg and Satinder Singh

***Abstract*—The rapid expansion of the Industrial Internet of Things (IIoT) necessitates the digitization of industrial processes in order to increase network efficiency. The integration of Digital Twin (DT) with IIoT digitizes physical objects into virtual representations to improve data analytics performance. Nevertheless, DT empowered IIoT generates a massive amount of data that is mostly sent to the cloud or edge servers for real-time analysis. However, unreliable public communication channels and lack of trust among participating entities causes various types of threats and attacks on the ongoing communication. Motivated from the aforementioned discussion, we present a blockchain and Deep Learning (DL) integrated framework for delivering decentralized data processing and learning in IIoT network. The framework first present a new DT model that facilitates construction of a virtual environment to simulate and replicate security-critical processes of IIoT. Second, we propose a blockchain-based data transmission scheme that uses smart contracts to ensure integrity and authenticity of data. Finally, the DL scheme is designed to apply the Intrusion Detection System (IDS) against valid data retrieved from blockchain. In DL scheme, a Long Short Term Memory-Sparse AutoEncoder (LSTMSAE) technique is proposed to learn the spatial-temporal representation. The extracted characteristics are further used by the proposed Multi-Head Self-Attention (MHSA)-based Bidirectional Gated Recurrent Unit (BiGRU) algorithm to learn long-distance features and accurately detect attacks. The practical implementation of our proposed framework proves considerable enhancement of communication security and data privacy in DT empowered IIoT network.**

***Index Terms*—Blockchain, Deep Learning (DL), Digital Twin (DT), Industrial Internet of Things (IIoT), Smart Contract.**

I. Introduction

THE Industrial Internet of Things (IIoT) is a network of intelligent interconnected industrial devices, and computing amenities deployed to achieve a highly efficient, autonomous, and improved manufacturing and industrial processes [1]. The success of IIoT depends on removing complexity from device deployment, connectivity, and management [2]. For example, IIoT allows the tracking of items as they transit from manufacturing to distribution in a supply chain. The rapid growth of IIoT has coincided with the introduction of cyber-attacks on vital infrastructure including smart factories, smart grids, and so on [3]. An attacker can use powerful techniques and tools to conduct malicious attacks, including Denial of Service (DoS), Distributed Denial of Service (DDoS), Man-in-the-Middle (MitM), firmware modification, false code injection and can take complete control of the IIoT infrastructure [4].

Existing traditional security solutions proposed in articles [5], [6], [7] designed various Intrusion Detection and Prevention System (IDS/IPS) but were frequently introduced after the asset became operational, rather during the initial design process. As a result, attackers can gather detailed knowledge of system behaviour and launch Advanced Persistent Threat (APT) attacks by exploiting vulnerabilities in the system infrastructure (e.g., smart grid management), that can even threaten public safety [8]. Additionally, due to the heterogeneous IIoT devices and the complicated industrial setting, making quick and intelligent decision in real-time is another challenging issue.

The Digital Twin (DT) is a new digitalization technology that generates a real-time digital simulation model of physical objects. In IIoT context, DT can assist researchers in running simulations to understand and analyze the behaviour of physical objects without actually manufacturing and deploying them [9]. The DTs look for data discrepancies between the physical, and virtual entities by collecting huge amount of data from all phases of the product life-cycle and provide simulation data to the physical entity so that it may improve its calibration and testing procedures [10]. Such recurrent processes improve DT models and their physical equivalents, allowing for more accurate estimate, prediction, and optimization of industrial operations. For instance, in smart grid management, DT collect data for power status from various types of sensors and present engineers with a virtual grid network layout to adapt real-time analysis in decision-making and execution [11].

The DT approaches proposed in the articles [12], [13], [14], [15], [16], [17] mostly used traditional cloud or edge-based architectures to map Cyber–Physical Systems (CPSs) to living digital models. However, DTs are data-driven and synchronization of real-time data needs a transparent and trustable solution among participating peers. Moreover, cloud/edge-based twins mandate trusting a third party, e.g., a cloud service provider, for IIoT data processing, which raises serious security and data

This work of Dr. Abhinav Kumar was supported in part by TiHAN Faculty Fellowship. (Corresponding authors: Prabhat Kumar; Sahil Garg.)

Prabhat Kumar is with the Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta 53850, Finland (Email: prabhat.kumar@lut.fi).

Randhir Kumar and Abhinav Kumar are with Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Hyderabad 502285, India (e-mail: randhir.honeywell@ieee.org, abhinavkumar@ee.iith.ac.in).

A. Antony Franklin is with the Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, Hyderabad 502285, India. (e-mail: antony.franklin@cse.iith.ac.in)

Sahil Garg is with the Resilient Machine Learning Institute (ReMI), Montreal, QC H3C 1K3, Canada (e-mail: garg.sahil1990@gmail.com)

Satinder Singh is with the Ultra I&C Communications, Montreal, QC H4T 1V7, ´ Canada (e-mail: Satinder.Singh@ultra-tcs.com)

privacy (e.g., performing various privacy attacking techniques, such as false-data injection, data poisoning and inference ones) concerns [18]. For example, a malicious cloud might expose or alter important industrial data without the owners authorization. Similarly, cloud owners (upto 90%) do not encrypt data before keeping it on their servers. Finally, a cloud or edge can be affected by a single point of failure [19]. As a solution to the above, blockchain and Deep Learning (DL) has emerged as a promising solution where the current study has been provided to ensure communication security and data privacy in DT empowered IIoT network.

Blockchain use cryptographic hashing algorithms and distributed consensus protocols to enable safe and secure data transfer [20]. The distributed ledgers of blockchain can help DTs in auditability, accessibility, and traceability of design data. The encrypted data of DTs stored in ledger can neither be changed nor can be controlled by a central authority. This functionality not only enables unparalleled levels of confidence and data integrity, but it also makes the DTs audit process more efficient and cost-effective [21]. Existing works presented in articles [22], [23], [24], [25], [26], [27] mainly abstracted blockchain as distributed and non-tampering ledger to store entire transaction, making blockchain quite inefficient and costly. Furthermore, only limited computational capability of locking scripts has been exploited.

Smart Contracts (SCs) are programmable logic that can be placed on a distributed network using modern blockchain technology. Functions and state variables are used by SCs to represent complicated business logic. Client requests are wrapped in transactions to invoke functions of SCs [28]. To keep state consistent, a primary node (also known as a miner) first assembles and executes a batch of SCs transactions in order, and then the remainder (known as validators) re-execute them serially in the same order. SCs provide high availability in the event of network node failures [29]. Furthermore, as their code is recorded on all nodes, it is immutable, making SC execution automatic, transparent, and the final output cryptographically verifiable by all participant nodes. Recent works presented in the articles [30], [31], [32] used SCs to manage data record or to provide access. However, in accordance to large-scale IIoT system their work are seriously limited in terms of scalability and flexibility. In the proposed framework InterPlenary File Systems (IPFS) platform is implemented as an off-chain storage system to provide high throughput and scalability during real-time data access by using minimal data storage costs [33].

The authenticated and valid data from blockchain technology can be further used to improve data analytic or utility model such as IDS performance. Deep Learning (DL) has become the mainstream technique to deal with unstructured, heterogeneous, and large volume of IIoT data. Although various DL-based IDS have been designed for attack identification and achieved better performance than traditional machine learning and statistical techniques [34]. There are two main limitations of existing approaches. First, the methods presented in [35], [36], [37] are purely based on supervised learning that rely on expert knowledge to manually label the attacks. On the other hand, IIoT network, necessitate a fast reaction time to human engagement. As a result, manual data labelling is challenging due to a lack of security expertise and a short response time. Second, many researchers in [38], [39] focus on using Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Convolution Neural Network (CNN) and other structures for attack classification. However, the length of IIoT network sequences are usually long and therefore these methods can not learn long dependent contextual association information.

A. System Model

This subsection presents a brief discussion on the applied DT empowered IIoT network and adversary model that are used to design and analyze the proposed framework.

1) Digital Twin Empowered IIoT Network: The proposed digital twin empowered IIoT network model is shown in Fig. 1, that consists of the following entities:

Trusted Authority (T_s): The T_s is a trustworthy entity having adequate computing and communication resources. The responsibility of T_s is to perform initialization of system parameters and registration of all different communicating entities prior to their placement in the network [28]. In addition, T_s also generates and provides certificates to each IIoT device and all edge/cloud nodes that include an identity, public key, and private key.

Engineer and Domain Knowledge (*EDK*): The *EDK* is used to gather, and provide information related to the system and network components of IIoT and is considered independent of any physical process. Furthermore, the data is generalized, specified once and then shared across multiple organizations. For example, industrial equipment manufacturers can provide device templates that defines safety and security policies of their devices. This can be used to derive the topological environment and logical connections among individual components of IIoT. Furthermore, when modeling a system, it is also vital to think about defining hierarchical relationships between components [40]. IIoT Twinning may also impose fine-grained regulations and restrictions across all hierarchical levels due to this modeling technique.

Generator (*GR*): The responsibility of *GR* is to convert the specification into a virtual environment. Initially, the information related to the specification is processed to retrieve the network, and devices topological structure and their associated security regulations. The virtual environment is then created by producing digital objects and imposing their properties. Finally, the parsed rules are kept in an abstract form for further analysis by a security module, namely IDS.

Digital Twin Model (*DTM*):The *DTM* is a core component of IIoT twinning with virtualized network infrastructure. It provides a realistic simulation environment for the physical processes and a runtime layout for virtual devices. Moreover, the generated virtualized IIoT environment is similar to its physical counterpart and provide various functionalities such as physical device types, their network protocols and control logic execution. The DT can be emulated by running the control logic, or simulated when a physical component has to be replicated. Once the virtual environment is generated

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3191601

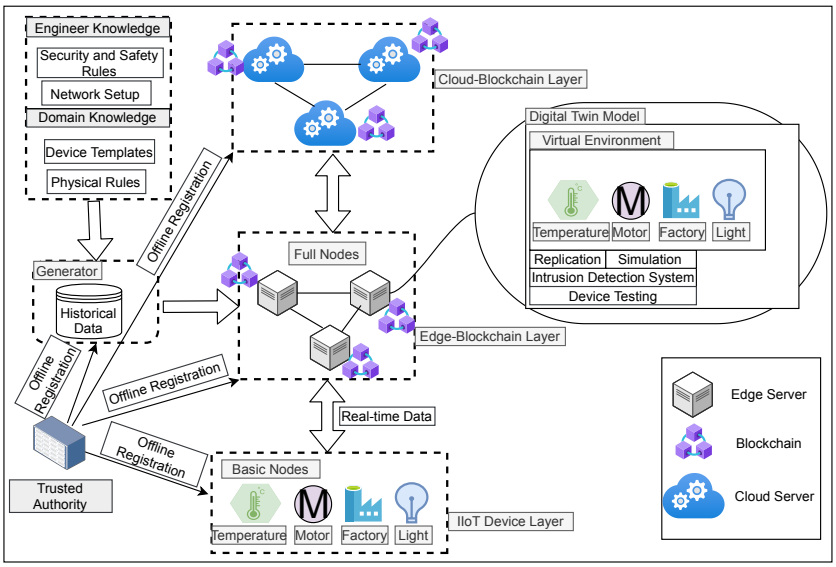


Fig. 1: The proposed framework for Digital Twin Empowered IIoT Network

based on the specification and configuration of DTs, we have two operation modes i.e., *simulation* and *replication* available. The digital twins run independently of their physical counterparts in *simulation* mode and enable users to examine process modifications, test equipment, and even improve manufacturing operations, similar to virtual commissioning. Additionally, security professionals can use this mode to conduct security test in a virtual environment, avoiding the risks associated with testing on a live system. On the other hand, the *replication* mode replicates data including log files, network connections, and sensor readings from the physical environment. Furthermore, as shown in Fig. 1, sensors or devices can be directly linked to the IIoT twinning architecture available at edge nodes.

IIoT Device Layer: This layer consist various heterogeneous IIoT devices denoted as *Basic Nodes (BNs)* (e.g., temperature sensors, hydraulic motors, light sensors) with limited computing resources. These devices are deployed in the industrial physical world to continuously gather and measure sensor data (transactions). The generated data is relayed to the edge-blockchain layer on a hop-by-hop forwarding basis.

Edge-Blockchain Layer (*EBL*): This layer consist powerful nodes i.e., *Full Nodes (FNs)* (e.g., edge-computing servers, industrial computer, data analysis server). As the IIoT devices are close to *EBL*, the edge-based design is more delay efficient than previous cloud-based IIoT architectures. Therefore, our proposed framework integrates DT with edge nodes, and are responsible for initial processing and creation of blocks in a tamper-resistant manner using smart contract-based consensus technique.

Cloud-Blockchain Layer (*CBL*): This layer consist various distributed cloud Servers *CS*, which can be multiple resource leasing platforms or websites. It is worth noting that the *CS* in our model are distributed and not controlled by a single entity. They build a cloud-based peer-to-peer network and share a significant quantity of historical blockchain data. Furthermore, each *CS* has a specific interface for receiving blockchain data from *EBL* nodes. When a *FN* storage space is full with blockchain data, it offloads the previous blockchain data to the *CBL*. Whenever the *CS* reach a consensus, it sends a "completed" message to the offloading node, which then contacts the other *FNs* to delete the specified blockchain data.

Intrusion Detection System (*IDS*): A DL-based *IDS* is designed to decide whether a particular traffic sample is an attack or normal in virtual environment of DT. Initially inputs to the IDS is data generated from *GR*. Later sensor data (transactions) from blockchain can be directly use to detect intrusive events. The main advantage of our approach is that the IDS provides a holistic security view of the physical process.

2) Adversary Model: In the adversary model we follow the widely accepted "Dolev-Yao (DY) threat model" [28] in DT empowered IIoT network. According to the DY model, an adversary named $\mathcal{A}$ can tamper information sent across an insecure (public) channel between any two participant (i.e., *GR* and *FN*, *BN* and *FN*, *FN* and *CS*). $\mathcal{A}$ is capable of not only eavesdropping on communications, but also of modifying, deleting, or injecting false messages into the communication channel. As a result, $\mathcal{A}$ can launch variety of attacks, including the replay attack, MitM attack, and impersonation attack. In addition to the DY model, we also consider "Canetti and Krawczyk's model (CK-adversary model)" [41], which is a more powerful model. In the CK-adversary model, an adversary $\mathcal{A}$ is capable of compromising secret credentials and hijacking session keys and session states in an ongoing session between two participants in the network. As a result, even if $\mathcal{A}$ hijacks a currently active session, he or she should not be able to jeopardize past or future session keys created between two entities. In addition, it is assumed that the *GR*, *FN*, *BN* and *CS* are semi-trusted entities.

B. Research Contribution

The following are the major contributions in this paper:

- We propose a new generalized architecture for integrating digital twin with IIoT edge servers that collects all industrial transaction records and thereby assist in improving communication security and data privacy in highly dynamic IIoT environment.
- A blockchain scheme is designed to securely transmit (without modification or deletion) IIoT data from *GR* or *BNs* to the *CS* by leveraging digital twin edge nodes. The authenticated data collected at *EBL* are used to create, validate and add blocks in the blockchain network with the help of smart contract-based "Proof-of-Authentication (PoA)" algorithm.
- The encrypted data is stored in IPFS-based off-chain storage system to minimize communication and computation overheads while ensuring scalability during real-time data access.
- A novel DL-based intrusion detection scheme using validated data obtained from blockchain is designed. The former first contains a Long Short Term Memory-Sparse AutoEncoder (LSTM-SAE)-based feature extraction technique to learn hidden feature structure and discriminative representations. The obtained spatial-temporal representation of IIoT traffic flows is forwarded to the proposed Multi-Head Self-Attention (MHSA)-based Bidirectional Gated Recurrent Unit (BiGRU) to recognize intrusive events.

The rest of this article is organized as follows. Section II discuss the detailed functional components of our proposed framework. The security analysis is performed in Section III. The experimental results are provided and evaluated in Section IV. Finally, Section V concludes this paper with future direction.

II. Proposed Framework for Secure Communication

A. Blockchain Scheme

1) System Initialization Phase: This section presents system initialization and assumed a trusted authority who is responsible for registering all the entity of network. The trusted authority (T_s) executes all the required parameters as discussed below.

Step-1: The T_s select a non-singular elliptic curve $E_{qn}(s,t)$ in the form $y^2 = x^3$+sx+t (mod q) in the galois field *gf(q)*, where *q* denotes large prime number over the condition $4s^3 + 27t^2 \neq 0$ (mod q) for non-singularity with ω as infinity point or zero point. Next, the T_s picks the base point $B \in E_{qn}$(s,t) with the order is as closest as of *q*, say *n* i.e., $n \,.\, B = \omega$, where $n \,.\, B$ denotes the scalar multiplicative elliptic curve point and $n \in Z_q$ denotes the discrete algorithm to the base point *B*.

Step-2: The T_s choose a one-way cryptographic hash function i.e., collision resistant, say *h(.)*. This can be computed using secure hash algorithm (SHA-256) for security reason, this provides 256-bit message digest.

Step-3: The T_s selects an identity ID_{T_s} and its master key M_{T_s} and also generates random private key $PR_{T_s} \in Z_q$, where $Z_q = \{1, 2, 3, 4, \ldots, q-1\}$. The T_s then computes public key as $PB_{T_s} = PR_{T_s} \,.\, B$.

Step-4: The T_s stores the PR_{T_s} and M_{T_s} as a secret keys and disseminates public parameters like $\{ E_{qn}(s,t)$, *B*, PB_{T_s}, *h(.)* $\}$.

2) Enrollment Phase: This phase discusses about the registration of each entities which is deployed over the network.

Cloud Server Enrollment: The T_s registers the cloud server (*CLS*) using following steps mentioned below.

Step-1: The T_s selects unique identity ID_{CLS} and evaluates the pseudo identity SID_{CLS}= h($ID_{T_s} \,||\, M_{T_s} \,||\, RT_{CLS}$), where RT_{CLS} denotes the enrollment timestamp of cloud server. Further, T_s selects temporal identity TID_{CLS} and a random secret $PR_{CLS} \in Z_q$ and finds the respective public key as $PB_{CLS} = PR_{CLS} \,.\, B$.

Step-2: The T_s creates a certificate for *CLS* as CRT_{CLS} = M_{T_s} + h ($PB_{T_s} \,||\, PB_{CLS} \,||\,$) * PR_{T_s} (mod q). Further, T_s preserve the cloud information i.e., (TID_{CLS}, SID_{CLS},CRT_{CLS}, PR_{CLS}, $E_{qn}(s,t)$, *h(.)*) into memory and shares the public key PB_{CLS} as public.

Edge Server Enrollment:

Step-1: The T_s selects unique identity ID_{EG} and evaluates the pseudo identity SID_{EG}= h($ID_{T_s} \,||\, M_{T_s} \,||\, RT_{EG}$), where RT_{EG} denotes the enrollment timestamp of edge server. Further, T_s selects temporal identity TID_{EG} and a random secret $PR_{EG} \in Z_q$ and finds the respective public key as $PB_{EG} = PR_{EG} \,.\, B$.

Step-2: The T_s creates a certificate for *EG* as CRT_{EG} = M_{T_s} + h ($PB_{T_s} \,||\, PB_{EG} \,||\,$) * PR_{T_s} (mod q). Further, T_s preserve the edge information i.e., (TID_{EG}, SID_{EG},CRT_{EG}, PR_{EG}, $E_{qn}(s,t)$, *h(.)*) into memory and shares the public key PB_{EG} as public.

Generator Enrollment:

Step-1: The T_s selects unique identity ID_{GN} and evaluates the pseudo identity SID_{GN}= h($ID_{T_s} \,||\, M_{T_s} \,||\, RT_{GN}$), where RT_{GN} denotes the enrollment timestamp of generator. Further, T_s selects temporal identity TID_{GN} and a random secret $PR_{GN} \in Z_q$ and finds the respective public key as $PB_{GN} = PR_{GN} \,.\, B$.

Step-2: The T_s creates a certificate for *GN* as CRT_{GN} = M_{T_s} + h ($PB_{T_s} \,||\, PB_{GN} \,||\,$) * PR_{T_s} (mod q). Further, T_s preserve the generator information i.e., (TID_{GN}, SID_{GN},CRT_{GN}, PR_{GN}, $E_{qn}(s,t)$, *h(.)*) into memory and shares the public key PB_{GN} as public.

IIoT node Enrollment:

Step-1: The T_s selects unique identity ID_{D_i} and evaluates the pseudo identity SID_{D_i}= h($ID_{T_s} \,||\, M_{T_s} \,||\, RT_{D_i}$), where RT_{D_i} denotes the enrollment timestamp of IIoT devices. Further, T_s selects temporal identity TID_{D_i} and a random secret $PR_{D_i} \in Z_q$ and finds the respective public key as PB_{D_i} = $PR_{D_i} \,.\, B$.

Step-2: The T_s creates a certificate for D_i as CRT_{D_i} = M_{T_s} + h ($PB_{T_s} \,||\, PB_{D_i} \,||\,$) * PR_{T_s} (mod q). Further, T_s preserve the IIoT device (D_i) information i.e., (TID_{D_i}, SID_{D_i},CRT_{D_i}, PR_{D_i}, $E_{qn}(s,t)$, *h(.)*) into memory and shares the public key PB_{D_i} as public.

3) Authentication Phase: This phase discusses authentication process of IIoT nodes to Edge Server, Generator to Edge Server, and Edge-server to cloud server. In this authentication process each entity maintains session key before making

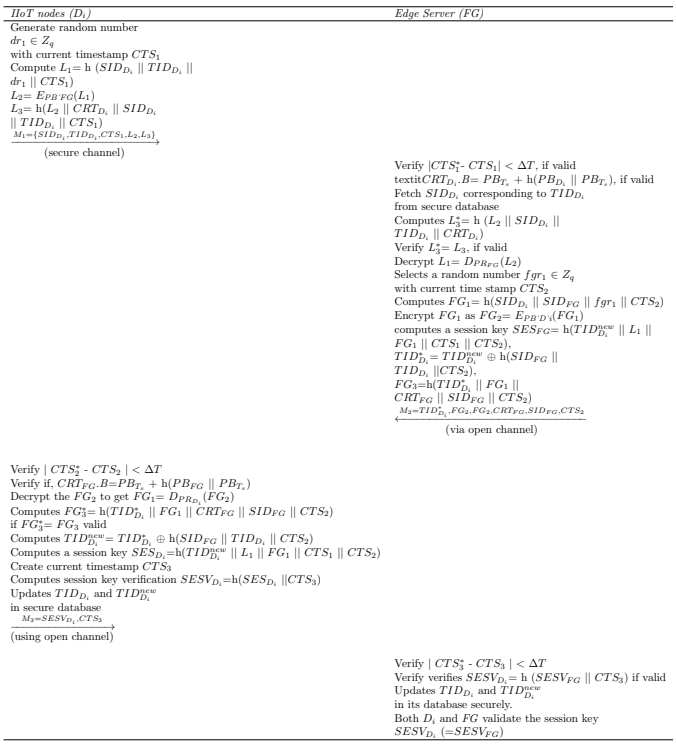


Fig. 2: Authentication Process between IIoT nodes and Edge Server

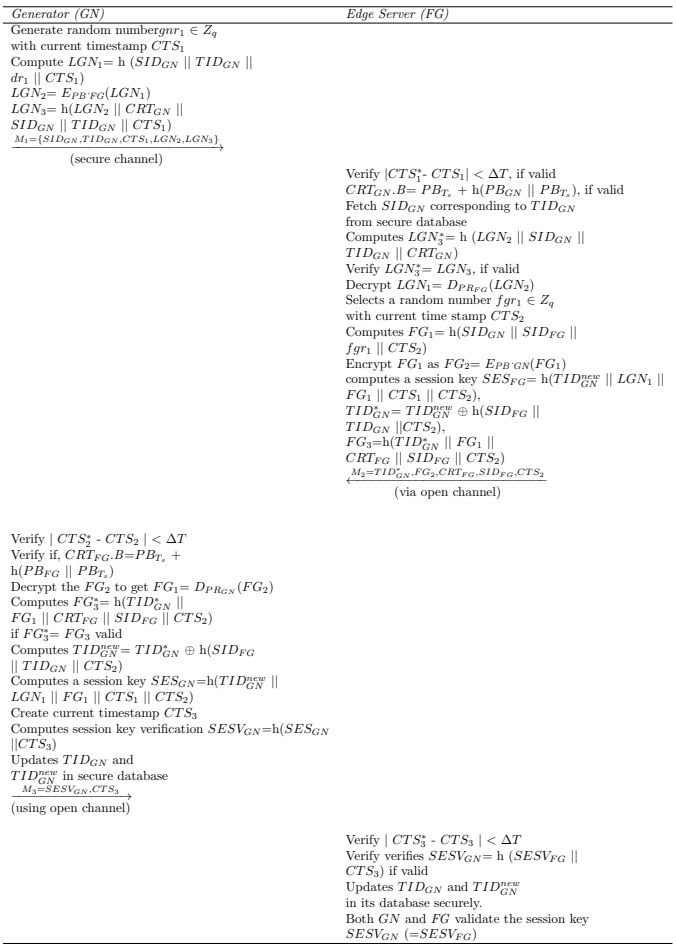


Fig. 3: Authentication Process between Generator and Edge Server

secure communications. This process ensures authorization of the entities in the network. These are the following steps needs to be executed to establish a session key during secure communication.

(i) IIoT nodes to Edge server Authentication

Step-1: D_i selects a random number $dr_1 \in Z_q$ with current timestamp CTS_1 and computes L_1= h (SID_{D_i} || TID_{D_i} || dr_1 || CTS_1). Next, D_i encrypt the L_1 as L_2= $E_{PB \cdot FG}(L_1)$. Further, D_i computes the L_3= h(L_2 || CRT_{D_i} || SID_{D_i} || TID_{D_i} || CTS_1) and generates a access request message M_1={SID_{D_i}, TID_{D_i}, CTS_1, L_2, L_3} and sent it to edge server using open channel.

Step-2: once the message M_1 is received at time CTS_1^*, edge server checks the timestamp $|CTS_1^*$- $CTS_1| < \Delta T$. If the timestamp is valid then edge server verifies certificate using CRT_{D_i}.*B*= PB_{T_s} + h(PB_{D_i} || PB_{T_s}) if it is also valid then edge server fetches SID_{D_i} corresponding to TID_{D_i} from secure database and computes L_3^*= h (L_2 || SID_{D_i} || TID_{D_i} || CRT_{D_i}) to check whether L_3^*= L_3. if it is valid then edge server decrypt L_2 as L_1= $D_{PR_{FG}}(L_2)$.

Step-3: Next, edge server selects a random number $fgr_1 \in Z_q$ with current time stamp CTS_2 and creates new temporary identity $TID_{D_i}^{new}$ and computes FG_1= h(SID_{D_i} || SID_{FG} || fgr_1 || CTS_2) and encrypt FG_1 as FG_2= $E_{PB \cdot D \cdot i}(FG_1)$. Further, edge server (*FG*) computes a session key SES_{FG}= h($TID_{D_i}^{new}$ || L_1 || FG_1 || CTS_1 || CTS_2), $TID_{D_i}^*$= $TID_{D_i}^{new}$ ⊕ h(SID_{FG} || TID_{D_i} ||CTS_2), and FG_3=h($TID_{D_i}^*$ || FG_1 || CRT_{FG} || SID_{FG} || CTS_2) and construct a reply message M_2= {$TID_{D_i}^*$, FG_2, FG_2, CRT_{FG}, SID_{FG}, CTS_2} and sent it to D_i using open channel.

Step-4: After receiving a reply message (M_2) from edge server at time CTS_2^*, D_i checks whether | CTS_2^* - CTS_2 | < ΔT is valid timestamp or not. if it is valid then D_i verifies certificate by CRT_{FG}.*B*=PB_{T_s} + h(PB_{FG} || PB_{T_s}). Next, D_i decrypt the FG_2 to get FG_1= $D_{PR_{D_i}}(FG_2)$. Further, D_i computes FG_3^*= h($TID_{D_i}^*$ || FG_1 || CRT_{FG} || SID_{FG} || CTS_2) and check, if FG_3^*= FG_3 then D_i computes $TID_{D_i}^{new}$= $TID_{D_i}^*$ ⊕ h(SID_{FG} || TID_{D_i} || CTS_2) and computes a session key SES_{D_i}=h($TID_{D_i}^{new}$ || L_1 || FG_1 || CTS_1 || CTS_2) and shares with *FG*. Next, D_i selects a current timestamp CTS_3 and computes session key verification $SESV_{D_i}$ using $SESV_{D_i}$=h(SES_{D_i} ||CTS_3) and updates the TID_{D_i} and $TID_{D_i}^{new}$ in secure database. Further, D_i creates an acknowledgment message M_3={$SESV_{D_i}$, CTS_3} and sent it to *FG* using open channel.

Step-5: After getting acknowledgment message M_3 at time CTS_3^*, then *FG* verifies the timestamp using | CTS_3^* - CTS_3 | < ΔT is valid timestamp or not. Next *FG* verifies $SESV_{D_i}$= h ($SESV_{FG}$ || CTS_3). If it matches successful, the *FG* establishes the session key $SESV_{D_i}$ (=$SESV_{FG}$) with D_i. At last, *FG* updates TID_{D_i} and $TID_{D_i}^{new}$ in its database securely. Fig.2 shows the entire authentication process between IIoT nodes (D_i) and Edge server (*FG*).

(ii) Generator to Edge server Authentication

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3191601

Step-1: GN selects a random number $gnr_1 \in Z_q$ with current timestamp CTS_1 and computes LGN_1= h (SID_{GN} || TID_{GN} || dr_1 || CTS_1). Next, D_i encrypt the LGN_1 as LGN_2= $E_{PB \cdot FG}(LGN_1)$. Further, GN computes the LGN_3= h(LGN_2 || CRT_{GN} || SID_{GN} || TID_{GN} || CTS_1) and generates a access request message M_1=$\{SID_{GN}$, TID_{GN}, CTS_1, LGN_2, $L_3\}$ and sent it to edge server using open channel.

Step-2: once the message M_1 is received at time CTS_1^*, edge server checks the timestamp $|CTS_1^*$- $CTS_1| < \Delta T$. If the timestamp is valid then edge server verifies certificate using CRT_{GN}.*B*= PB_{T_s} + h(PB_{GN} || PB_{T_s}) if it is also valid then edge server fetches SID_{GN} corresponding to TID_{GN} from secure database and computes LGN_3^*= h (LGN_2 || SID_{GN} || TID_{GN} || CRT_{GN}) to check whether LGN_3^*= LGN_3. if it is valid then edge server decrypt LGN_2 as LGN_1= $D_{PR_{FG}}(LGN_2)$.

Step-3: Next, edge server selects a random number fgr_1 $\in Z_q$ with current time stamp CTS_2 and creates new temporary identity TID_{GN}^{new} and computes FG_1= h(SID_{GN} || SID_{FG} || fgr_1 || CTS_2) and encrypt FG_1 as FG_2= $E_{PB \cdot GN}(FG_1)$. Further, edge server (*FG*) computes a session key SES_{FG}= h(TID_{GN}^{new} || LGN_1 || FG_1 || CTS_1 || CTS_2), TID_{GN}^*= TID_{GN}^{new} $\oplus$ h(SID_{FG} || TID_{GN} ||CTS_2), and FG_3=h(TID_{GN}^* || FG_1 || CRT_{FG} || SID_{FG} || CTS_2) and construct a reply message M_2= $\{TID_{GN}^*$, FG_2, FG_2, CRT_{FG}, SID_{FG}, $CTS_2\}$ and sent it to GN using open channel.

Step-4: After receiving a reply message (M_2) from edge server at time CTS_2^*, GN checks whether $|$ CTS_2^* - CTS_2 $| < \Delta T$ is valid timestamp or not. if it is valid then GN verifies certificate by CRT_{FG}.*B*=PB_{T_s} + h(PB_{FG} || PB_{T_s}). Next, GN decrypt the FG_2 to get FG_1= $D_{PR_{GN}}(FG_2)$. Further, GN computes FG_3^*= h(TID_{GN}^* || FG_1 || CRT_{FG} || SID_{FG} || CTS_2) and check, if FG_3^*= FG_3 then GN computes TID_{GN}^{new}= TID_{GN}^* $\oplus$ h(SID_{FG} || TID_{GN} || CTS_2) and computes a session key SES_{GN}=h(TID_{GN}^{new} || LGN_1 || FG_1 || CTS_1 || CTS_2) and shares with *FG*. Next, GN selects a current timestamp CTS_3 and computes session key verification $SESV_{GN}$ using $SESV_{GN}$=h(SES_{GN} ||CTS_3) and updates the TID_{GN} and TID_{GN}^{new} in secure database. Further, GN creates an acknowledgment message M_3=$\{SESV_{GN}$, $CTS_3\}$ and sent it to *FG* using open channel.

Step-5: After getting acknowledgment message M_3 at time CTS_3^*, then *FG* verifies the timestamp using $|$ CTS_3^* - CTS_3 $| < \Delta T$ is valid timestamp or not. Next *FG* verifies $SESV_{GN}$= h ($SESV_{FG}$ || CTS_3). If it matches successful, the *FG* establishes the session key $SESV_{GN}$ (=$SESV_{FG}$) with GN. At last, *FG* updates TID_{GN} and TID_{GN}^{new} in its database securely. Fig.3 shows the entire authentication process between Generator (GN) and Edge server (*FG*). *(iii) Edge server to cloud server Authentication*

Step-1: FG selects a random number $fgr_1 \in Z_q$ with current timestamp CTS_1 and computes LFG_1= h (SID_{FG} || TID_{FG} || fgr_1 || CTS_1). Next, FG encrypt the LFG_1 as LFG_2= $E_{PB \cdot CLS}(LFG_1)$. Further, FG computes the LFG_3= h(LFG_2 || CRT_{FG} || SID_{FG} || TID_{FG} || CTS_1) and generates a access request message M_1=$\{SID_{FG}$, TID_{FG},

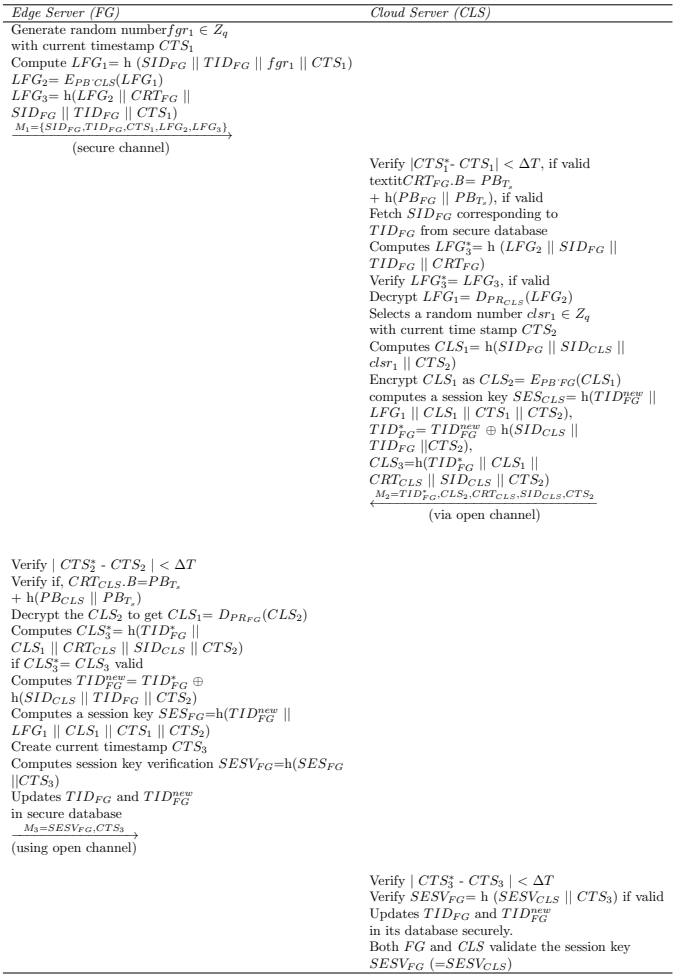


Fig. 4: Authentication Process between Edge Server and Cloud Server

CTS_1, LFG_2, $LFG_3\}$ and sent it to cloud server using open channel.

Step-2: once the message M_1 is received at time CTS_1^*, cloud server checks the timestamp $|CTS_1^*$- $CTS_1| < \Delta T$. If the timestamp is valid then cloud server verifies certificate using CRT_{FG}.*B*= PB_{T_s} + h(PB_{FG} || PB_{T_s}) if it is also valid then edge server fetches SID_{FG} corresponding to TID_{FG} from secure database and computes LFG_3^*= h (LFG_2 || SID_{FG} || TID_{FG} || CRT_{FG}) to check whether LFG_3^*= LFG_3. if it is valid then cloud server decrypt LFG_2 as LFG_1= $D_{PR_{CLS}}(LFG_2)$.

Step-3: Next, cloud server selects a random number $clsr_1$ $\in Z_q$ with current time stamp CTS_2 and creates new temporary identity TID_{FG}^{new} and computes CLS_1= h(SID_{FG} || SID_{CLS} || $clsr_1$ || CTS_2) and encrypt CLS_1 as CLS_2= $E_{PB \cdot FG}(CLS_1)$. Further, cloud server (*FG*) computes a session key SES_{CLS}= h(TID_{FG}^{new} || LFG_1 || CLS_1 || CTS_1 || CTS_2), TID_{FG}^*= TID_{FG}^{new} $\oplus$ h(SID_{CLS} || TID_{FG} ||CTS_2), and CLS_3=h(TID_{FG}^* || CLS_1 || CRT_{CLS} || SID_{CLS} || CTS_2) and construct a reply message M_2= $\{TID_{FG}^*$, CLS_2, CRT_{CLS}, SID_{CLS}, $CTS_2\}$ and sent it to FG using open channel.

Step-4: After receiving a reply message (M_2) from cloud

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3191601

Algorithm 1 Smart contracts enabled Proof-of-Authentication

Input: Transactions (Tx_i), public key (PB_{D_i}), IIoT nodes (D_i)
Output: Verification and Block creation .
function verify(D_i, Tx_i, PB_{D_i}, CRT_{D_i})
assert($VERIFY(PB_{D_i}$, $IIoTNonce[message.sender]$, $CRT_{D_i})$)
assert (*Authenticate*($SESV_{D_i}$))
emit Event(message.sender, Tx_i, "Allow")
end function
function $VERIFY(Tx_i$, nonce,$CRT_{D_i})$
$Message_{Signed}$=message.sender || nonce || Tx_i
if ($Message_{Signed}$, CRT_{D_i} == owner) then
return true
else
return false
end if
end function
function *Authenticate*($SESV_{D_i}$)
policies= $session_{verify}[SESV_{D_i}]$
if policies is successful then
Block is created with Details such as TX_{BLOCK}= $\{Tx_i^{hash}$, ID_{D_i}, CRT_{D_i}, PB_{D_i}, $Block_{Hash}$, CTS_{D_i}, nonce$\}$
Finally, block is committed
end function

server at time CTS_2^*, FG checks whether $\mid CTS_2^*$ - $CTS_2 \mid < \Delta T$ is valid timestamp or not. if it is valid then FG verifies certificate by CRT_{CLS}.*B*=PB_{T_s} + h($PB_{CLS} \mid\mid PB_{T_s}$). Next, FG decrypt the CLS_2 to get CLS_1= $D_{PR_{FG}}(CLS_2)$. Further, FG computes CLS_3^*= h($TID_{FG}^* \mid\mid CLS_1 \mid\mid CRT_{CLS} \mid\mid SID_{CLS} \mid\mid CTS_2$) and check, if CLS_3^*= CLS_3 then FG computes TID_{FG}^{new}= $TID_{FG}^* \oplus$ h($SID_{CLS} \mid\mid TID_{FG} \mid\mid CTS_2$) and computes a session key SES_{FG}=h($TID_{FG}^{new} \mid\mid LFG_1 \mid\mid CLS_1 \mid\mid CTS_1 \mid\mid CTS_2$) and shares with *CLS*. Next, FG selects a current timestamp CTS_3 and computes session key verification $SESV_{FG}$ using $SESV_{FG}$=h($SES_{FG} \mid\mid CTS_3$) and updates the TID_{FG} and TID_{FG}^{new} in secure database. Further, FG creates an acknowledgment message M_3=$\{SESV_{FG}, CTS_3\}$ and sent it to *CLS* using open channel.

Step-5: After getting acknowledgment message M_3 at time CTS_3^*, then *CLS* verifies the timestamp using $\mid CTS_3^*$ - $CTS_3 \mid < \Delta T$ is valid timestamp or not. Next *CLS* verifies $SESV_{FG}$= h ($SESV_{CLS} \mid\mid CTS_3$). If it matches successful, the *CLS* establishes the session key $SESV_{FG}$ (=$SESV_{CLS}$) with FG. At last, *CLS* updates TID_{FG} and TID_{FG}^{new} in its database securely. Fig.4 shows the entire authentication process between Edge Server (FG) and Cloud server (CLS).

4) Smart Contract Verification and Block Addition Phase: This phase includes verification of D_i using its certificate CRT_{D_i} and $SESV_{D_i}$ based authentication. The verification of D_i is approached using smart contract based Proof-of-Authentication (PoA). The detail of authentication process is detailed in the Algorithm 1.

B. Deep Learning Scheme

1) Development of LSTMSAE-based Feature Extraction Technique: The LSTMSAE is a combination of LSTM and SAE. In fact, LSTMSAE is an AE with sparseness penalty

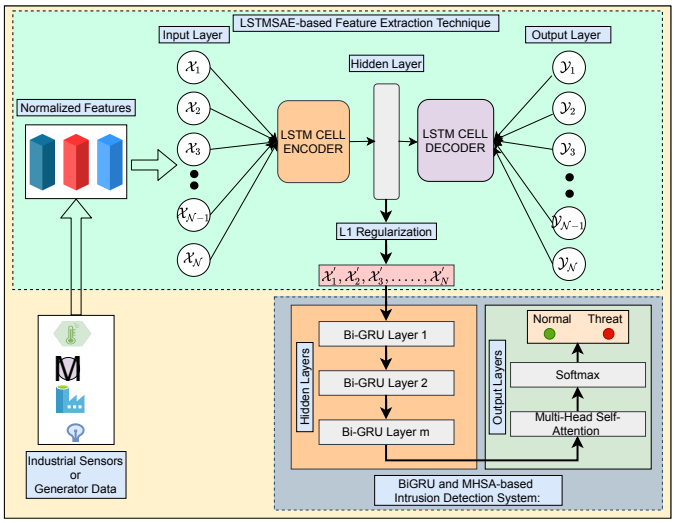


Fig. 5: The working architecture of proposed deep-learning scheme for intrusion detection

item that uses LSTM to extract features. The extracted feature from the industrial sensors or generator data is used by the proposed BiGRU with MH-SA-based IDS to detect intrusion. Assume there are *m* sensors in an industrial setting and each sensor collects $\mathcal{N}$ samples, the input variable matrix can be denoted as $\mathcal{X}$, i.e., $\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2, \ldots\ldots, \mathcal{X}_\mathcal{N}] \in \mathbb{R}^{\mathcal{N}\times m}$. The output matrix of the hidden layer of a LSTMSAE-based feature extraction technique is $\mathring{\mathcal{A}}$, i.e., $\mathring{\mathcal{A}} = \left[\mathring{\mathcal{A}}_1, \mathring{\mathcal{A}}_2, \ldots\ldots, \mathring{\mathcal{A}}_\mathcal{N}\right] \in \mathbb{R}^{\mathcal{N}\times d}$, where d is the dimension of the feature vector in the hidden layer.

Long Short-Term Memory (LSTM:) The LSTM solves "vanishing gradient" problem that classic Recurrent Neural Networks (RNNs) have during back propagation. In the first stage, the LSTM structure generates decision vectors and picks candidate data. The values for these vectors lies between 0 and 1, where LSTM ignores vectors close to 0 and retains vectors with values close to 1. Specifically, LSTM produce the input gate $\mathcal{I}$ by using previous LSTM units hidden state $\mathcal{H}_{\mathcal{T}-1}$ and input $\mathcal{X}_\mathcal{T}$ of current unit at step $\mathcal{T}$.

$$\mathcal{I}_\mathcal{T} = \sigma\left(\mathcal{W}_\mathcal{I}\left[\mathcal{H}_{\mathcal{T}-1}, \mathcal{X}_\mathcal{T}\right] + \mathcal{B}_\mathcal{I}\right). \tag{1}$$

The activation function is denoted by σ, the weighted matrix is $\mathcal{W}_\mathcal{I}$, and the bias between two connected components is $\mathcal{B}_\mathcal{I}$. In order to assess if the prior unit state $\mathcal{C}_{\mathcal{T}-1}$ should be kept as the current unit state, LSTM use forget gate $\mathcal{F}_\mathcal{T}$ with $\mathcal{H}_{\mathcal{T}-1}$ and $\mathcal{X}_\mathcal{T}$ as two input values.

$$\mathcal{F}_\mathcal{T} = \sigma\left(\mathcal{W}_\mathcal{F}\left[\mathcal{H}_{\mathcal{T}-1}, \mathcal{X}_\mathcal{T}\right] + \mathcal{B}_\mathcal{F}\right). \tag{2}$$

where the forget gates weight and bias matrices are denoted by $\mathcal{W}_\mathcal{F}$, and $\mathcal{B}_\mathcal{F}$ respectively. The $\mathcal{I}_\mathcal{T}$ use $\mathcal{X}_\mathcal{T}$ and $\mathcal{H}_{\mathcal{T}-1}$ and is responsible to update the information in cell state $\widetilde{\mathcal{C}_\mathcal{T}}$.

$$\widetilde{\mathcal{C}_\mathcal{T}} = \tanh\left(\mathcal{W}_\mathcal{C}\left[\mathcal{H}_{\mathcal{T}-1}, \mathcal{X}_\mathcal{T}\right] + \mathcal{B}_\mathcal{C}\right). \tag{3}$$

The current $\mathcal{C}_\mathcal{T}$ is connected to the previous $\mathcal{C}_{\mathcal{T}-1}$ and the input candidate $\mathcal{C}_\mathcal{T}$ is computed as

$$\mathcal{C}_\mathcal{T} = F_\mathcal{T} * \mathcal{C}_{\mathcal{T}-1} + I_\mathcal{T} * \widetilde{\mathcal{C}_\mathcal{T}} \tag{4}$$

Finally, LSTM use an output gate $\mathcal{O}_\mathcal{T}$ to identify the next timesteps hidden state $\mathcal{H}_\mathcal{T}$. The $\mathcal{H}_\mathcal{T}$ contains information about the past stages, which is used to create predictions. A two-step approach is used to obtain the $\mathcal{H}_\mathcal{T}$ for the subsequent timestep:

$$\mathcal{O}_\mathcal{T} = \sigma\left(\mathcal{W}_\mathcal{O}\left[\mathcal{H}_{\mathcal{T}-1}, \mathcal{X}_\mathcal{T}\right] + \mathcal{B}_\mathcal{O}\right). \tag{5}$$

$$\mathcal{H}_\mathcal{T} = \mathcal{O}_\mathcal{T} * \tanh\left(\mathcal{C}_\mathcal{T}\right) \tag{6}$$

where the output gates weight and bias matrices are represented as $\mathcal{W}_\mathcal{O}$ and $\mathcal{B}_\mathcal{O}$, respectively.

LSTM Sparse AutoEncoder (LSTMSAE): The SAE is a form of AE that consist an encoder and a decoder. The SAE use the input input $\mathcal{X}$ to obtain a low-dimensional or latent pattern $\mathcal{X}^{'}$, whereas the decoder use the hidden layer features to reconstruct the input $\mathcal{X}$. The encoding and decoding formula is computed as

$$\mathcal{X}^{'} = \mathcal{S}\left(\mathcal{W}_1\mathcal{X} + \mathcal{B}_1\right) \tag{7}$$

$$\widehat{\mathcal{X}} = \mathcal{S}\left(\mathcal{W}_2\mathcal{X}^{'} + \mathcal{B}_2\right) \tag{8}$$

where $\widehat{\mathcal{X}}$ is the output reconstruction of $\mathcal{X}$. $\mathcal{W}_1$, $\mathcal{B}_1$ and $\mathcal{W}_2$ $\mathcal{B}_2$ denotes weight matrix and bias vectors of encoding and decoding layer, respectively. The nonlinear activation functions, namely ReLU, sigmoid and tanh are denoted by $\mathcal{S}(\bullet)$. The AE model parameters $(\mathcal{W}_1, \mathcal{W}_2, \mathcal{B}_1, \mathcal{B}_2)$ can be learnt from a training set by reducing the following objective:

$$\begin{aligned}\left\{\widehat{\mathcal{W}}_1, \widehat{\mathcal{W}}_2, \widehat{\mathcal{B}}_1, \widehat{\mathcal{B}}_2\right\} &= \underset{W_1,W_2,B_1,B_2}{arg\,min}\left\{\mathcal{L}_{\mathcal{AE}}\left(\mathcal{W}, \mathcal{B}\right)\right\} \\ &= \underset{W_1,W_2,B_1,B_2}{arg\,min}\left\{\frac{1}{2\mathcal{N}}\sum_{i=1}^{\mathcal{N}}\mathcal{L}\left(\mathcal{X}_i, \widehat{\mathcal{X}}_i\right)\right. \\ &\left. + \frac{\lambda}{2}\sum_{\mathcal{R}=1}^{2}\left\|\mathcal{W}_\mathcal{R}\right\|_\mathcal{F}^2\right\}\end{aligned} \tag{9}$$

where $\mathcal{N}$ denotes the total training samples and the hidden layer size is denoted by $\mathcal{R}$ for the *i*th training sample $\mathcal{X}_i$ and its associated reconstruction output $\widehat{\mathcal{X}}_i$. The error function is denoted by $\mathcal{L}(\bullet)$ and use cross entropy. The term λ improves the generalization capability of the model and act as a regularization parameter. The Kullback Leibler (KL) divergence function is used as the sparse constraint in SAE, which is a kind of stacked AE. The KL divergence is calculated as follows:

$$\sum_{\mathcal{J}=1}^{m}\mathrm{KL}\left(\rho \parallel \hat{\rho}_\mathcal{J}\right) = \sum_{\mathcal{J}=1}^{m}\left\{(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_\mathcal{J}} + \rho\log\frac{\rho}{\hat{\rho}_\mathcal{J}}\right\} \tag{10}$$

where the sparsity parameter is denoted by ρ and for all training samples in the *j*th neuron the average activation value is denoted as $\hat{\rho}_\mathcal{J}$. By adding a sparsity penalty term η to the AE objective function we get SAE and is represented as

$$\mathcal{L}_{SAE}\left(\mathcal{W}_1, \mathcal{W}_2, \mathcal{B}_1, \mathcal{B}_2\right) = \mathcal{L}_{AE}\left(\mathcal{W}, \mathcal{B}\right) + \eta\sum_{\mathcal{J}=1}^{m}\mathrm{KL}\left(\rho \parallel \hat{\rho}_\mathcal{J}\right) \tag{11}$$

where ρ is frequently a minimal integer, such as 0.05, to obtain a sparse representation. In the LSTMSAE model, the LSTM network is integrated with the SAE, which implies LSTM handles the encoding and decoding, as illustrated in Fig. 5. By constraining the latent space to be smaller in dimensionality than the input, the LSTMSAE is forced to learn the most important aspects of the training data.

2) Design of MHSA-based BiGRU for Intrusion Detection System: In order to detect intrusion from the features extracted by the hidden layer of the LSTMSAE technique, a BiGRU is adapted as base model. Then, a Multi-Head Self-Attention (MHSA) mechanism is introduced to capture long IIoT traffic sequences. Finally, a feed-forward layer with the softmax function is used to predict the probabilities for each class present in the dataset. The output of LSTMSAE technique is sequence of patterns $\mathring{\mathcal{A}}$, i.e., $\mathring{\mathcal{A}} = \left[\mathring{\mathcal{A}}_1, \mathring{\mathcal{A}}_2, \ldots\ldots, \mathring{\mathcal{A}}_\mathcal{N}\right]$. The ultimate production at time $\mathcal{T}$ is decided by the preceding and next frames at time $\mathcal{T}-1$ and $\mathcal{T}+1$, respectively, in a BiGRU arrangement. To be more specific, one GRU computes the forward hidden state $\overrightarrow{\mathcal{H}_1}, \overrightarrow{\mathcal{H}_2}, \ldots, \overrightarrow{\mathcal{H}_\mathcal{N}}$), while the other computes the backward hidden state $\overleftarrow{\mathcal{H}_1}, \overleftarrow{\mathcal{H}_2}, \ldots, \overleftarrow{\mathcal{H}_\mathcal{N}}$). The final BiGRU output is then calculated as a concatenated vector of forward hidden state outputs and backward processes, where the $\rightarrow$ and $\leftarrow$ indicates the forward and backward processes, respectively. The following are the BiGRU transition functions in hidden units:

$$\overrightarrow{\mathcal{H}_\mathcal{T}} = \mathcal{F}\left(\overrightarrow{\mathring{\mathcal{A}}_\mathcal{T}}, \overrightarrow{\mathcal{H}_{\mathcal{T}-1}}; \overrightarrow{\Theta_{GRU}}\right)$$

$$= \begin{cases}\overrightarrow{\mathcal{U}_\mathcal{T}} = \sigma\left(\overrightarrow{\mathcal{W}_\mathcal{U}} \bullet \left[\overrightarrow{\mathcal{H}_{\mathcal{T}-1}}, \overrightarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overrightarrow{\mathcal{B}_\mathcal{U}}\right), \\ \overrightarrow{\mathcal{R}_\mathcal{T}} = \sigma\left(\overrightarrow{\mathcal{W}_\mathcal{R}} \bullet \left[\overrightarrow{\mathcal{H}_{\mathcal{T}-1}}, \overrightarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overrightarrow{\mathcal{B}_\mathcal{R}}\right), \\ \overrightarrow{\mathcal{C}_\mathcal{T}} = \tanh\left(\overrightarrow{\mathcal{W}_\mathcal{C}} \bullet \left[\overrightarrow{\mathcal{R}_\mathcal{T}} * \overrightarrow{\mathcal{H}_{\mathcal{T}-1}}, \overrightarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overrightarrow{\mathcal{B}_\mathcal{C}}\right), \\ \overrightarrow{\mathcal{F}_\mathcal{T}} = \left(\overrightarrow{1-\mathcal{U}_\mathcal{T}}\right) * \overrightarrow{\mathcal{H}_{\mathcal{T}-1}} + \overrightarrow{\mathcal{U}_\mathcal{T}} * \overrightarrow{\mathcal{C}_\mathcal{T}}.\end{cases} \tag{12}$$

$$\overleftarrow{\mathcal{H}_\mathcal{T}} = \mathcal{F}\left(\overleftarrow{\mathring{\mathcal{A}}_\mathcal{T}}, \overleftarrow{\mathcal{H}_{\mathcal{T}+1}}; \overleftarrow{\Theta_{GRU}}\right)$$

$$= \begin{cases}\overleftarrow{\mathcal{U}_\mathcal{T}} = \sigma\left(\overleftarrow{\mathcal{W}_\mathcal{U}} \bullet \left[\overleftarrow{\mathcal{H}_{\mathcal{T}+1}}, \overleftarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overleftarrow{\mathcal{B}_\mathcal{U}}\right), \\ \overleftarrow{\mathcal{R}_\mathcal{T}} = \sigma\left(\overleftarrow{\mathcal{W}_\mathcal{R}} \bullet \left[\overleftarrow{\mathcal{H}_{\mathcal{T}+1}}, \overleftarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overleftarrow{\mathcal{B}_\mathcal{R}}\right), \\ \overleftarrow{\mathcal{C}_\mathcal{T}} = \tanh\left(\overleftarrow{\mathcal{W}_\mathcal{C}} \bullet \left[\overleftarrow{\mathcal{R}_\mathcal{T}} * \overleftarrow{\mathcal{H}_{\mathcal{T}+1}}, \overleftarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right] + \overleftarrow{\mathcal{B}_\mathcal{C}}\right), \\ \overleftarrow{\mathcal{F}_\mathcal{T}} = \left(\overleftarrow{1-\mathcal{U}_\mathcal{T}}\right) * \overleftarrow{\mathcal{H}_{\mathcal{T}+1}} + \overleftarrow{\mathcal{U}_\mathcal{T}} * \overleftarrow{\mathcal{C}_\mathcal{T}}.\end{cases} \tag{13}$$

where $[\bullet, \bullet]$ represents connection between two vectors. $\overrightarrow{\mathcal{U}_\mathcal{T}}$, $\overrightarrow{\mathcal{R}_\mathcal{T}}$, $\overrightarrow{\mathcal{C}_\mathcal{T}}$, $\overrightarrow{\mathcal{F}_\mathcal{T}}$ and $\overleftarrow{\mathcal{U}_\mathcal{T}}$, $\overleftarrow{\mathcal{R}_\mathcal{T}}$, $\overleftarrow{\mathcal{C}_\mathcal{T}}$, $\overleftarrow{\mathcal{F}_\mathcal{T}}$, denotes update gate, reset gate, candidate cell and final state for the forward and backward process, respectively. The parameters for the forward and backward phases are $\overrightarrow{\Theta_{GRU}}$ and $\overleftarrow{\Theta_{GRU}}$, respectively, which are shared across all time steps and learnt during model training. $\bullet$ denotes two matrices multiplied by their elements, $*$ is the dot product operation of matrices. $\overrightarrow{\mathcal{W}_\mathcal{U}}$, $\overrightarrow{\mathcal{W}_\mathcal{R}}$, $\overrightarrow{\mathcal{W}_\mathcal{C}}$ and $\overleftarrow{\mathcal{W}_\mathcal{U}}$, $\overleftarrow{\mathcal{W}_\mathcal{R}}$, $\overleftarrow{\mathcal{W}_\mathcal{C}}$ denotes parameter matrix from $\left[\overrightarrow{\mathcal{H}_{\mathcal{T}-1}}, \overrightarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right]$ to $\overrightarrow{\mathcal{U}_\mathcal{T}}$, $\overrightarrow{\mathcal{R}_\mathcal{T}}$ $\overrightarrow{\mathcal{C}_\mathcal{T}}$ for forward pass and $\left[\overleftarrow{\mathcal{H}_{\mathcal{T}+1}}, \overleftarrow{\mathring{\mathcal{A}}_\mathcal{T}}\right]$ to $\overleftarrow{\mathcal{W}_\mathcal{U}}$, $\overleftarrow{\mathcal{W}_\mathcal{R}}$, $\overleftarrow{\mathcal{W}_\mathcal{C}}$ for backward pass. $\overrightarrow{\mathcal{B}_\mathcal{U}}$, $\overrightarrow{\mathcal{B}_\mathcal{R}}$, $\overrightarrow{\mathcal{B}_\mathcal{C}}$ and $\overleftarrow{\mathcal{B}_\mathcal{U}}$, $\overleftarrow{\mathcal{B}_\mathcal{R}}$, $\overleftarrow{\mathcal{B}_\mathcal{C}}$ are the bias weights for forward and backward process. σ and $\tanh$ are non-linear activation function of *Sigmoid*$(\bullet)$ and *Tanh*$(\bullet)$. In summary, BiGRU hidden element representation

$\mathcal{H}_\mathcal{T}$ represents the concatenation of output produced from forward and backward methods.

$$\mathcal{H}_\mathcal{T} = \overrightarrow{\mathcal{H}_\mathcal{T}} \oplus \overleftarrow{\mathcal{H}_\mathcal{T}} \tag{14}$$

The hidden state representation vector obtained from BiLSTM layer $\mathcal{H}^\mathcal{A} = (\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_\mathcal{T})$ is then sent to the MHSA layer. MHSA consist three linear blocks for query, key and value. Each linear block is made up of $\mathcal{M}$ separate linear layers. Here, $\mathcal{M}$ is the total number of heads. The MHSA is introduced to extract and learn long-term dependency patterns from the input traffic sequence i.e., $\mathcal{H}^\mathcal{A}$, that applies linear transformation and creates $\mathcal{Q}_i$, $\mathcal{K}_i$, $\mathcal{V}_i$ using *i*th linear layers. Where $i = [1, 2, \ldots, \mathcal{Z}]$. where $\mathcal{Z}$ denotes total number of attention heads. The $\mathcal{Q}_i$, $\mathcal{K}_i$, $\mathcal{V}_i$ are fed into scaled dot product attention layer. For the *i*th head, the scaled dot product attention $\mathcal{A}_i$ is as follows:

$$\mathcal{A}_i = \text{Softmax}\left(\frac{{\mathcal{K}_i}^\mathcal{T}\mathcal{Q}_i}{\sqrt{d_q}}\right)\mathcal{V}_i \tag{15}$$

The query vector's dimension is $\sqrt{d_q}$. Using basic concatenation, we aggregate the attention output from all of the heads and input it into the feed-forward layer having softmax function.

$$\mathcal{M} = \text{Concat}\left(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_i, \ldots, \mathcal{A}_\mathcal{Z}\right)\mathcal{W}_\mathcal{O} \tag{16}$$

where $\mathcal{A}_i$ is a $d_q \times \mathcal{T}$ dimensional matrix. The final output attention matrix $\mathcal{M}$ from the multi-head attention block will have $\mathcal{Z} \times d_q \times \mathcal{T}$ matrix dimensions. Since the Concat(•) operation is applied to the feature dimension of all the matrices. Finally, in the last layer of proposed IDS we use *softmax* function to identify attack and normal instances. Let us assume that the MHSA block produces output denoted as $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\mathcal{T})$ using which the *softmax* function φ generates network outcome as one-hot encoded $\mathcal{C}$-dimensional vector *y*. Then, we can determine the probability *p* of a single input $\mathcal{M}$ belonging to a particular attack class (*y*) using below Eq.

$$p\left(\widehat{\mathcal{Y}}_c = \mathcal{Y}_c \mid \mathcal{M}\right) = \varphi\left(\mathcal{M}\right)\mathcal{Y}_c = \frac{\exp^{\mathcal{M}_c}}{\sum_{d=1}^{\mathcal{C}} \exp^{\mathcal{M}_d}} \quad (\mathcal{C} = 1, 2, \ldots, c) \tag{17}$$

In order to compute loss for each prediction at each timestamp we use $\mathcal{C}$-way cross-entropy loss that gives the probability across $\mathcal{C}$ class labels using below Eq.

$$\mathcal{LOSS} = \frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}}\sum_{c=1}^{\mathcal{C}} \mathcal{Y}_{ic} \log\left(\widehat{\mathcal{Y}}_{ic}\right) \tag{18}$$

where $\mathcal{N}$ represents the batch size, $\mathcal{C}$ represents the number of classes, $\mathcal{Y}$ and $\widehat{\mathcal{Y}}$, represent the actual and predicted class labels, respectively.

III. Security Analysis

Security analysis against different attack is discussed in this section.

1) Message Authentication: The T_s validates a pseudo identity (SID_{D_i}) of IIoT nodes ID_{D_i} over the created message M_1=$\{SID_{D_i}, TID_{D_i}, CTS_1, L_2, L_3\}$ while authentication. Thus, adversary cann't create same message and signature during a certain time interval.

2) Privacy-Preservation: The IIoT nodes share a message $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$ = (TID_{D_i} || CRT_{D_i} || RT_{D_i} || PB_{D_i}) and pseudo identity SID_{D_i}= h(ID_{T_s} || M_{T_s} || RT_{D_i}), where RT_{D_i} denotes the enrollment timestamp of IIoT devices. Next certificate is created CRT_{D_i} = M_{T_s} + h (PB_{T_s} || PB_{D_i} ||) * PR_{T_s} (mod q). As a result, knowing the true identification of IIoT nodes, the attacker must complete this action within a particular time frame. This procedure guarantees that the system's privacy is protected.

3) Replay Attack: The IIoT nodes share a message $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$ = (TID_{D_i} || CRT_{D_i} || RT_{D_i} || PB_{D_i}) and pseudo identity SID_{D_i}= h(ID_{T_s} || M_{T_s} || RT_{D_i}). The T_s validates this message with PB_{D_i} and RT_{D_i}. This entire computation process prevent against valid message broadcasting to unauthorized IIoT nodes. from unauthorized IIoT nodes ID_{D_i} and thus this process prevents from replay attack.

4) Man-in-the-Middle (MitM) Attack: The message created by the IIoT nodes, i.e., $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$ = $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$ = (TID_{D_i} || CRT_{D_i} || RT_{D_i} || PB_{D_i}) and pseudo identity SID_{D_i}= h(ID_{T_s} || M_{T_s} || RT_{D_i}), where RT_{D_i} denotes the enrollment timestamp of IIoT devices with authorized certificate CRT_{D_i} for the sent message $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$. Thus, this process defend against MitM attacks.

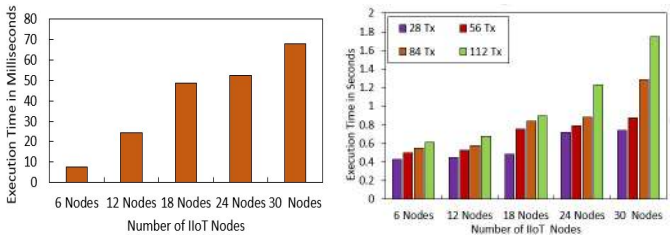
5) Impersonation Attack: To make a impersonation attack, attacker must generates a message $\mathcal{D}_{\rangle_{\Updownarrow\int\}}}$ = (TID_{D_i} || CRT_{D_i} || RT_{D_i} || PB_{D_i}) and pseudo identity SID_{D_i}= h(ID_{T_s} || M_{T_s} || RT_{D_i}) with authorized certificates CRT_{D_i} and timestamp RT_{D_i}. The exact message, certificates, and timestamp creation is highly impossible to match. Thus, this computational process prevent from an impersonation attack.

IV. Performance Analysis

This section evaluates the performance of proposed blockchain and deep learning framework in DT empowered IIoT network for security and privacy.

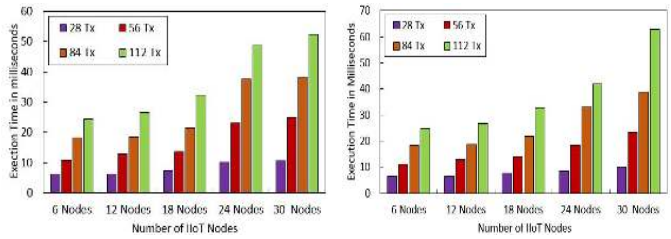
A. Experimental Setup

The simulations are carried on a Tyrone Windows 10 PC, 128 GB RAM and featuring an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz (2 processors), and a 2 TB hard disk. Keras API of Tensorflow was used to implement deep learning approaches and scikit-learn library to implement machine learning techniques. We used Ganache and Ethereum to design a private blockchain and implement smart contract. WEB3 Provider interface of Ethereum was used to connect both blockchain networks. We deploy an Interplanetary File System (IPFS) version 0.4.19 to store IIoT transactions. We extensively experimented on two different network datasets CICIDS-2017 and ToN-IoT denoted as $\mathcal{D}_\mathcal{M}$ and $\mathcal{D}_\mathcal{N}$, respectively. To speed up convergence during training, we preprocessed datasets and performed a feature scaling step mentioned in [28], [29]. We divided the dataset into two subgroups



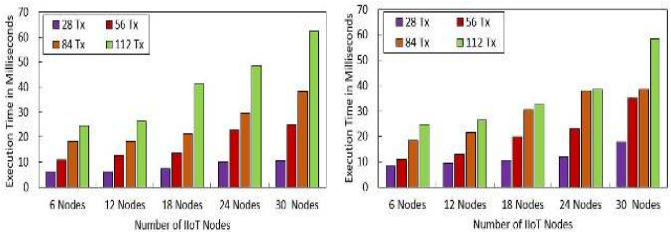
(a) Time analysis of registration by varying IIoT nodes

(b) Upload time analysis by varying Tx in IPFS-based off-chain storage layer



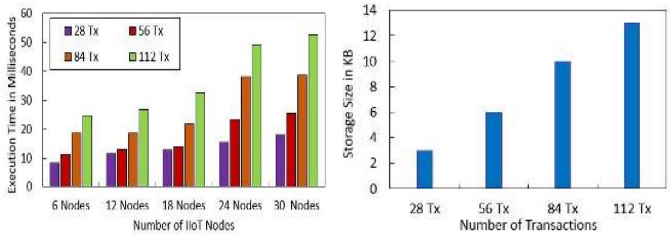
(c) PoA based consensus for different Tx

(d) Time analysis for block creation by varying Tx



(e) Time analysis for block access by varying Tx

(f) Time analysis for contract deployment by varying Tx



(g) Time analysis for non-repudiation using digital signature creation by varying Tx

(h) IPFS-based off-chain storage consumption (in KB) for actual data

Fig. 6: Analysis of blockchain scheme

to assess model performance in the train-test-split evaluation. The first subset, referred to as the "training dataset," fitted the model, while the second, referred to as the "testing dataset," evaluated the model. Finally, the efficiency is proposed IDS is evaluated using five popular evaluation metrics, namely ACcuracy ($\mathcal{AC}$), PRecision ($\mathcal{PR}$), Detection Rate ($\mathcal{DR}$), $\mathcal{F}1$ and False Alarm Rate ($\mathcal{FAR}$) as discussed in [34].

B. Numerical Results of Blockchain Scheme

The privacy and security in the proposed framework is maintained using IIoT nodes registration and its authentication. The registration and authentication process is performed against the malicious behavior of nodes in the network. The analysis of registration process is shown in Fig. 6a. The actual transactions

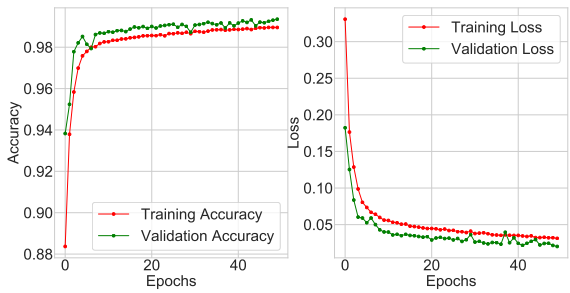


Fig. 7: Accuracy vs loss computed with the LSTMSAE method on the $\mathcal{D}_{\mathcal{N}}$ dataset

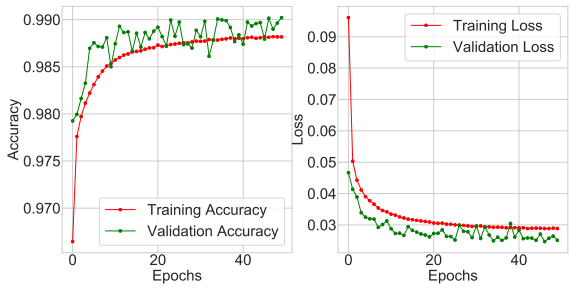


Fig. 8: Accuracy vs loss computed with the LSTMSAE method on the $\mathcal{D}_{\mathcal{M}}$ dataset

upload with IPFS storage layer is shown in Fig. 6b. Fig. 6c, Fig 6d, and Fig 6e shows block mining, block creation, and block access time, respectively. It can be noted from the figures that, the execution time increases with increase in IIoT nodes in the network. The Fig. 6f and Fig. 6g illustrates digital signature and contract deployment time. The digital signature enables non-repudiation in the entire network. Fig. 6h depicts actual transaction storage size in KB. The IPFS storage layer is used to calculate the size of different numbers of transactions. It is also seen that as the number of transactions grows, the storage size grows.

C. Numerical Results of Deep Learning Scheme

The results of DL scheme are discussed in this subsection. Initially, the proposed IDS uses *Adam* optimizer with 0.0005 *learning rate* and a *mini-batch* size of 128 for 50 *epochs*. Fig 7 and Fig 8 shows $\mathcal{AC}$ vs loss obtained on $\mathcal{D}_{\mathcal{N}}$ and $\mathcal{D}_{\mathcal{M}}$ dataset, respectively. We see that the training and validation $\mathcal{AC}$ gradually increases together indicating that the trained model is not having a variance problem and can be effectively generalized on the testing dataset. The training and validation $\mathcal{AC}$ of the $\mathcal{D}_{\mathcal{N}}$ dataset grows progressively and converges at 99.01% and 99.12%, respectively and also the loss reduces consistently and converges at 0.0210% and 0.0201%, respectively. Similarly, the training and validation $\mathcal{AC}$ progressively rises and converges at 99.82% and 99.92%, respectively, while the loss steadily reduces and converges at 0.0291% and 0.0243%, respectively, with the $\mathcal{D}_{\mathcal{M}}$ dataset. Table I and Table II report the class-wise experimental results for each attack and normal classes in terms of $\mathcal{PR}$, $\mathcal{DR}$, $\mathcal{F}1$-score and $\mathcal{FAR}$ using $\mathcal{D}_{\mathcal{N}}$ and $\mathcal{D}_{\mathcal{M}}$ datasets, respectively. It

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3191601

TABLE I: Class-wise (%) results for proposed IDS using $\mathcal{D}_{\mathcal{N}}$ dataset.

Parameters	Backdoor	DDoS	DoS	Injection	MITM	Normal	Password	Ransomware	Scanning	XSS
$\mathcal{PR}$	99.90	97.43	99.96	95.39	89.06	100.00	97.41	99.48	99.51	95.82
$\mathcal{DR}$	99.94	96.81	98.50	95.12	89.78	100.00	98.45	99.70	99.19	99.20
$\mathcal{F}1$	99.41	97.55	98.73	95.75	89.41	100.00	98.71	99.02	99.20	98.41
$\mathcal{FAR}$	0.00013	0.00211	0.00056	0.00201	0.00032	0.00000	0.00019	0.00014	0.00011	0.00232

TABLE II: Class-wise (%) results for proposed IDS using $\mathcal{D}_{\mathcal{M}}$ dataset

Parameters	BENIGN	DoS Hulk	DDoS	PortScan	DoS GoldenEye	FTPPatator	DoS slowloris	DoS Slowhttptes	SSHPatator	Bot	Web Attack
$\mathcal{PR}$	98.36	89.52	98.16	87.12	84.22	90.19	97.26	89.25	99.15	99.71	94.23
$\mathcal{DR}$	98.74	99.89	94.50	89.99	95.14	71.43	97.52	96.24	91.41	36.81	02.19
$\mathcal{F}1$	98.13	94.21	95.18	72.89	89.24	82.89	98.72	94.14	95.58	58.82	04.66
$\mathcal{FAR}$	0.0957211	0.008810	0.000714	0.001561	0.000630	0.000021	0.000032 0	0.000014	0.000011	0.00010	0.00001

TABLE III: Comparison of multi-vector $\mathcal{DR}$ (%) with other baselines using $\mathcal{D}_{\mathcal{N}}$ dataset

Techniques	Backdoor	DDoS	DoS	Injection	MITM	Normal	Password	Ransomware	Scanning	XSS
NB	99.22	26.80	91.70	92.96	95.11	100.00	75.32	79.98	96.91	19.02
DT	100.00	100.00	100.00	0.00	0.00	100.00	100.00	100.00	100.00	100.00
RF	99.98	90.40	91.97	93.53	0.00	100.00	97.81	99.40	95.74	85.47
Proposed IDS	99.94	96.81	98.50	95.12	89.78	100.00	98.45	99.70	99.19	99.20

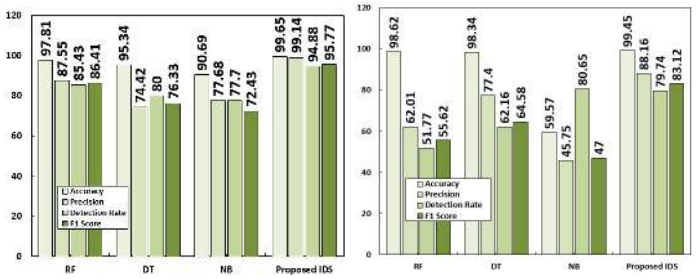
TABLE IV: Comparison of multi-vector $\mathcal{DR}$ (%) with other baselines using $\mathcal{D}_{\mathcal{M}}$ dataset

Techniques	BENIGN	DoS Hulk	DDoS	PortScan	DoS GoldenEye	FTPPatator	DoS slowloris	DoS Slowhttptes	SSHPatator	Bot	Web Attack
RF	100.00	95.00	100.00	97.00	50.00	72.00	0.00	55.00	0.00	0.00	0.00
DT	100.00	90.00	99.00	97.00	66.00	99.00	35.00	0.00	97.00	0.00	0.00
NB	55.00	89.00	98.00	50.00	99.00	100.00	60.00	77.00	97.00	76.00	08.00
Proposed IDS	98.74	99.89	94.50	89.99	95.14	71.43	97.52	96.24	91.41	36.81	02.19

is seen that the proposed IDS has obtained higher numerical values for these metrics and has lowered $\mathcal{FAR}$ close to 0%.

D. Comparative Analysis

The performance of DL-based IDS is compared with three contemporary ML techniques namely, Nave Bayes (NB), Decision Tree (DT) and Random Forest (RF). First, we use class-wise $\mathcal{DR}$ based on $\mathcal{D}_{\mathcal{N}}$ and $\mathcal{D}_{\mathcal{M}}$ datasets. It can be observed in Table III and Table IV the $\mathcal{DR}$ is better for most of the classes compared with other techniques. An efficient IDS has high values of $\mathcal{AC}$, $\mathcal{PR}$, $\mathcal{DR}$, and $\mathcal{F}1$. The obtained values for these parameters are shown in Fig 9a and Fig. 9b. It is seen that the proposed IDS has achieved 99.65%, 99.14%, 94.88%, 95.77% values for above parameters with $\mathcal{D}_{\mathcal{N}}$ and 99.45%, 88.16%, 79.74%, 83.12% with $\mathcal{D}_{\mathcal{M}}$ datasets, respectively. The values are high compared to RF, DT and NB. The ability of DL model (i.e., integration of LSTMSAE and MHSA-based BiGRU) to simulate the spatial-temporal representations inherent in DT empowered IIoT network data can justify this performance. In addition, the integration of blockchain in the network has helped in preventing malicious or low-quality parameters allowing the IDS to be more efficient and trustworthy than other competitors.



(a) Comparison based on $\mathcal{D}_{\mathcal{N}}$ dataset (b) Comparison based on $\mathcal{D}_{\mathcal{M}}$ dataset

Fig. 9: Comparison with baseline techniques

V. Conclusion with Future Directions

In this paper, we introduced a novel digital twin-enabled IIoT network. We first presented the digital twin empowered system model for IIoT network that includes IIoT devices, edge servers, and cloud servers. We developed a blockchain and deep learning integrated framework in the context of digital twin empowered IIoT system that provides data privacy and offers secure data communication. The blockchain features such as, transparency, decentralization, and immutability effectively ensure the access control functions dependability and auditability. We conducted extensive analysis on the Ethereum test network to illustrate the scalability and effectiveness of blockchain scheme. Additionally, we incorporated IPFS off-chain storage system to store encrypted IIoT transactions. Finally, extensive data-driven simulations using deep learning architecture show that we can take full advantage of blockchain scheme to achieve the highest detection rate and classification accuracy. The future research work will focus on fine-grained credentials (read, write, execute, delegate, and so on), as well as privacy-preservation (attribute-based signature and zero knowledge proof), and the integration of federated learning in digital twin-enabled networks.

References

[1] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[2] S. Latif, M. Driss, W. Boulila, S. S. Jamal, Z. Idrees, J. Ahmad *et al.*, "Deep learning for the industrial internet of things (iiot): A comprehensive survey of techniques, implementation frameworks, potential applications, and future directions," *Sensors*, vol. 21, no. 22, p. 7518, 2021.

[3] H. Yao, P. Gao, P. Zhang, J. Wang, C. Jiang, and L. Lu, "Hybrid intrusion detection system for edge-based iiot relying on machine-learning-aided detection," *IEEE Network*, vol. 33, no. 5, pp. 75–81, 2019.

[4] G. Falco, C. Caldera, and H. Shrobe, "Iiot cybersecurity risk modeling for scada systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4486–4495, 2018.

[5] K. Yu, L. Tan, S. Mumtaz, S. Al-Rubaye, A. Al-Dulaimi, A. K. Bashir, and F. A. Khan, "Securing critical infrastructures: Deep-learning-based threat detection in iiot," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 76–82, 2021.

[6] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[7] A. R. Javed, S. u. Rehman, M. U. Khan, M. Alazab, and T. R. G, "Canintelliids: Detecting in-vehicle intrusion attacks on a controller area network using cnn and attention-based gru," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1456–1466, 2021.

[8] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan, and M. S. Haghighi, "Anomaly detection in automated vehicles using multistage attention-based convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4291–4300, 2021.

[9] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[10] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[11] A. Saad, S. Faddel, T. Youssef, and O. A. Mohammed, "On the implementation of iot-based digital twin for networked microgrids resiliency against cyber attacks," *IEEE transactions on smart grid*, vol. 11, no. 6, pp. 5138–5150, 2020.

[12] J. Liu, S. Zhang, H. Liu, and Y. Zhang, "Distributed collaborative anomaly detection for trusted digital twin vehicular edge networks," in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2021, pp. 378–389.

[13] M. Groshev, C. Guimarães, J. Martín-Pérez, and A. de la Oliva, "Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence," *IEEE Communications Magazine*, vol. 59, no. 8, pp. 14–20, 2021.

[14] W. Tärneberg, M. Gunnarsson, M. Kihl, and C. Gehrmann, "A cloud-native digital twin with adaptive cloud-based control and intrusion detection," *Electronic Communications of the EASST*, vol. 80, 2021.

[15] W. Tärneberg, P. Skarin, C. Gehrmann, and M. Kihl, "Prototyping intrusion detection in an industrial cloud-native digital twin," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1. IEEE, 2021, pp. 749–755.

[16] Q. Qi, D. Zhao, T. W. Liao, and F. Tao, "Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing," in *International Manufacturing Science and Engineering Conference*, vol. 51357. American Society of Mechanical Engineers, 2018, p. V001T05A018.

[17] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86 769–86 777, 2019.

[18] H. R. Hasan, K. Salah, R. Jayaraman, M. Omar, I. Yaqoob, S. Pesic, T. Taylor, and D. Boscovic, "A blockchain-based approach for the creation of digital twins," *IEEE Access*, vol. 8, pp. 34 113–34 126, 2020.

[19] A. Khan, F. Shahid, C. Maple, A. Ahmad, and G. Jeon, "Toward smart manufacturing using spiral digital twin framework and twinchain," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1359–1366, 2022.

[20] P. Kumar, R. Kumar, G. Srivastava, G. P. Gupta, R. Tripathi, T. R. Gadekallu, and N. N. Xiong, "Ppsf: A privacy-preserving and secure framework using blockchain-based machine-learning for iot-driven smart cities," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2326–2341, 2021.

[21] I. Yaqoob, K. Salah, M. Uddin, R. Jayaraman, M. Omar, and M. Imran, "Blockchain for digital twins: Recent advances and future research challenges," *IEEE Network*, vol. 34, no. 5, pp. 290–298, 2020.

[22] M. Keshk, B. Turnbull, N. Moustafa, D. Vatsalan, and K.-K. R. Choo, "A privacy-preserving-framework-based blockchain and deep learning for protecting smart power networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5110–5118, 2019.

[23] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2020.

[24] Q. N. Tran, B. P. Turnbull, H.-T. Wu, A. de Silva, K. Kormusheva, and J. Hu, "A survey on privacy-preserving blockchain systems (ppbs) and a novel ppbs-based framework for smart agriculture," *IEEE Open Journal of the Computer Society*, vol. 2, pp. 72–84, 2021.

[25] O. Alkadi, N. Moustafa, and B. Turnbull, "A collaborative intrusion detection system using deep blockchain framework for securing cloud networks," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2020, pp. 553–565.

[26] W. Liang, L. Xiao, K. Zhang, M. Tang, D. He, and K.-C. Li, "Data fusion approach for collaborative anomaly intrusion detection in blockchain-based systems," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[27] S. Suhail, R. Hussain, R. Jurdak, and C. S. Hong, "Trustworthy digital twins in the industrial internet of things with blockchain," *IEEE Internet Computing*, pp. 1–1, 2021.

[28] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, and N. Kumar, "P2sf-iov: A privacy-preservation-based secured framework for internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[29] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, T. R. Gadekallu, and G. Srivastava, "Sp2f: a secured privacy-preserving framework for smart agricultural unmanned aerial vehicles," *Computer Networks*, vol. 187, p. 107819, 2021.

[30] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1594–1605, 2019.

[31] M. Yutaka, Y. Zhang, M. Sasabe, and S. Kasahara, "Using ethereum blockchain for distributed attribute-based access control in the internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[32] J. Hao, C. Huang, W. Tang, Y. Zhang, and S. Yuan, "Smart contract-based access control through off-chain signature and on-chain evaluation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[33] R. Kumar and R. Tripathi, "Towards design and implementation of security and privacy framework for internet of medical things (iomt) by leveraging blockchain and ipfs technology," *The Journal of Supercomputing*, pp. 1–40, 2021.

[34] P. Kumar, G. P. Gupta, and R. Tripathi, "Tp2sf: A trustworthy privacy-preserving secured framework for sustainable smart cities by leveraging blockchain and machine learning," *Journal of Systems Architecture*, vol. 115, p. 101954, 2021.

[35] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.

[36] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. den Hartog, "Ton˙iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion datasets," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[37] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in iot networks," *IEEE Access*, vol. 9, pp. 103 906–103 926, 2021.

[38] M. M. Hassan, S. Huda, S. Sharmeen, J. Abawajy, and G. Fortino, "An adaptive trust boundary protection for iiot networks using deep-learning feature-extraction-based semisupervised model," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2860–2870, 2020.

[39] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[40] M. Eckhart and A. Ekelhart, "Towards security-aware virtual environments for digital twins," in *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*, ser. CPSS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 61–72. [Online]. Available: https://doi.org/10.1145/3198458.3198464

[41] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2002, pp. 337–351.

Prabhat Kumar received his Ph.D. degree in Information Technology, National Institute of Technology Raipur, Raipur, India, under the prestigious fellowship of Ministry of Human Resource and Development (MHRD) funded by the Government of India in 2022. Thereafter, he worked with Indian Institute of Technology Hyderabad, India as a Post-Doctoral Researcher under project "Development of Indian Telecommunication Security Assurance Requirements for IoT devices". He is currently working as Post-Doctoral Researcher with the Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta, Finland. He has many research contributions in the area of Machine Learning, Deep Learning, Federated Learning, Big Data Analytics, Cybersecurity, Blockchain, Cloud Computing, Internet of Things and Software Defined Networking. He has authored or coauthored over 20+ publications in high-ranked journals and conferences. One of his Ph.D. publication was recognized as a top cited article by WILEY in 2020-21.

Randhir Kumar received his Ph.D. degree in Information Technology, National Institute of Technology Raipur, Raipur, India in 2021. He is currently working as Post-Doctoral Researcher with the Department of Electrical Engineering, Indian Institute of Technology Hyderabad, India. He has published his research article in leading journal and conferences from IEEE, Elsevier, Springer, and John Wiley. He has published more than 40 research article in the reputed journals and conferences. His paper has been published in some of the high impact factor journals such as – IEEE Internet of Things, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Network Science and Engineering, IEEE Transactions on Green Communications and Networking, IEEE Transactions on Industrial Informatics, IEEE COMSNETs, IEEE ICC, Computer Networks, JPDC, and Transactions on Emerging Telecommunications Technologies (ETT Wiley). He has qualified UGC-NET in the year of 2018. He has been awarded as Best Engineer Trainee by Honeywell Technology, India in the year 2009. His research interest includes cryptographic techniques, information security, blockchain technology, and web mining. He is also an IEEE Member.

Abhinav Kumar (Senior Member, IEEE) received the B.Tech., M.Tech., and Ph.D. degrees in electrical engineering from the Indian Institute of Technology Delhi, India, in 2009 and 2013, respectively. From September to November, 2013, he was a Research Associate with the Indian Institute of Technology Delhi. From December 2013 to November 2014, he was a Postdoctoral Fellow at the University of Waterloo, Canada. Since November 2014, he has been with the Indian Institute of Technology Hyderabad, India, where he is currently an Associate Professor. His research interests include wireless communications and networking.

Antony Franklin (Senior Member, IEEE) is an associate professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Hyderabad (IITH), India. He received his B.E. in electronics and communication engineering from Madurai Kamaraj University, India, in 2000, an M.E. in computer science and engineering from Anna University, India, in 2002, and Ph.D. in computer science and engineering from the Indian Institute of Technology Madras, India, in 2010. He was a senior engineer at the DMC R&D Center, Samsung Electronics, South Korea between 2012 and 2015, and a research engineer at the Electronics and Telecommunications Research Institute (ETRI), South Korea between 2010 and 2012. His current research is on the development of next generation mobile network architectures and protocols, which includes Cloud Radio Access Networks (C-RAN), Mobile Edge Computing (MEC), Multi-Radio Aggregation, Internet of Things (IoT), and SDN/NFV. He has published over 50 articles in refereed international journals and conferences.

Sahil Garg (S'15, M'18) received the Ph.D. degree from the Thapar Institute of Engineering and Technology, Patiala, India, in 2018. He is currently a Research Associate at Resilient Machine Learning Institute (ReMI) co-located with ÃL'cole de Technologie SupÃl'rieure (ÃL'TS), MontrÃl'al. Prior to this, he worked as a Postdoctoral Research Fellow at ÃL'TS, Montreal and MITACS Researcher at Ericsson, Montreal. He has many research contributions in the area of Machine Learning, Big Data Analytics, Knowledge Discovery, Cloud Computing, Internet of Things, Software Defined Networking, and Vehicular Ad-hoc Networks. He has over 80 publications in high ranked Journals and Conferences, including 50+ top-tier journal papers and 30+ reputed conference articles. He was the recipient of the prestigious Visvesvaraya PhD fellowship from the Ministry of Electronics & Information Technology under Government of India (2016-2018). He has been awarded the 2021 IEEE Systems Journal Best Paper Award; the 2020 IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher) and the IEEE ICC best paper award in 2018 at Kansas City, Missouri. He is currently a Managing Editor of Springer's Humancentric Computing and Information Sciences (HCIS) journal; and an Associate Editor of IEEE Network Magazine, IEEE Transactions on Intelligent Transportation Systems, Elsevier's Applied Soft Computing (ASoC), and Wiley's International Journal of Communication Systems. (IJCS).

Satinder Singh is a data science director with REMI Resilient Machine learning Institute and Director advanced system with Ultra electronics Communication. He received a B.Eng from École de Technologie Supérieur (2003) related to wireless communications, digital signal processing. At Ultra communication, his earlier work involved development of wireless communications hardware and waveforms for tactical backhaul network presently deployed with US army TRILOS program. Since taking leadership of REMI, his interests are aligned with AI/ML development for modern tactical network: situational and spectrum awareness, development of AI/ML empowered waveforms, AI/ML assisted Auto-PACE, Ease of Use and end user acceptance of Machine learning algorithms.