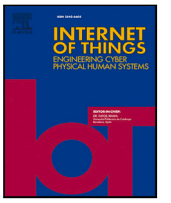
Internet of Things 19 (2022) 100575

Contents lists available at ScienceDirect

Internet of Things

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



Research article

Autoencoder based Consensus Mechanism for Blockchain-enabled Industrial Internet of Things

Murshedul Arifeen [a], Tapotosh Ghosh [b], Rakibul Islam [c], Akm Ashiquzzaman [d], Juncheol Yoon [d], Jinsul Kim [d,*]

[a] School of Computing, Robert Gordon University, Aberdeen, Scotland, United Kingdom
[b] Department of Computer Science and Engineering, United International University, Dhaka, Bangladesh
[c] Department of ICT, Bangladesh University of Professionals, Dhaka, Bangladesh
[d] Department of ICT Convergence System Engineering, Chonnam National University, Gwangju 61186, South Korea

ARTICLE INFO

Keywords:
IoT
Autoencoder
Blockchain
Hyperledger
Security

ABSTRACT

Conventional blockchain technologies developed for cryptocurrency applications involve complex consensus algorithms which are not suitable for resource constrained Internet of Things (IoT) devices. Therefore, several lightweight consensus mechanisms that are suitable for IoT devices have been proposed in recent studies. However, these lightweight consensus mechanisms do not verify the originality of the data generated by the IoT devices, so false and anomalous data may pass through and be stored in the ledger for further analysis. In this work to address the data originality verification problem, we propose an autoencoder (AE)-integrated Chaincode (CC)-based consensus mechanism in which the AE differentiates normal data from anomalous data. The AE is invoked through the CC once a transaction is initiated; the result returned from the AE to the CC is stored in the ledger. We have conducted a case study to train and test the AE model on the IoTID20 dataset. Also, Minifabric (MF) is used to implement the CC and illustrate the CC operation that stores only original IoT data. Moreover, the performance has been shown for the CC in terms of latency and throughput.

1. Introduction

The Industrial Internet of Things (IIoT) refers to the application of IoT networks in the field of industrial applications and smart manufacturing processes [1] and is the key technology of industry 4 [2]. IoT devices equipped with various types of sensors and actuators [3] are attached to industrial machines or engines and form a wireless network by connecting with each other through high-bandwidth (BW) communication links. These tiny IoT devices sense the intended data, collect useful information and periodically share these data with the central hub. Utilizing this type of IoT network in an industrial building or smart factory enhances the manufacturing process or production process [4]. It also enables efficient controlling and monitoring of industrial machinery. A central authority or admin can analyze the data collected from the sensors and use the results to increase the effectiveness of industrial operations through a centralized process like cloud architecture. The IIoT encompasses a wide range of applications, including automated and remote equipment management and monitoring, predictive maintenance, faster implementation of improvements, pinpoint inventories, quality control, supply chain optimization, and plant safety improvement [5,6].

However, integration of an IoT network with cloud-based architecture is subject to several limitations and shortcomings. For example, cloud-based IoT architectures are vulnerable to several security threats and suffer from various bottlenecks, such as

* Corresponding author.
E-mail address: jsworld@chonnam.ac.kr (J. Kim).

https://doi.org/10.1016/j.iot.2022.100575
Received 14 July 2021; Received in revised form 4 July 2022; Accepted 4 July 2022
Available online 8 July 2022

single point of failure [7]. Security and privacy attacks can damage an entire IoT network and disrupt normal network operations. Moreover, an IoT network is complex, heterogeneous, distributed and unattended, leaving it vulnerable to various types of attacks [8]. The data that is exchanged among IoT devices may be modified or tempered by a nefarious actor or attacker [9,10]. Tempering of the data hampers proper monitoring of the industrial process and causes incorrect information to be passed along to the authority. Therefore, it is vital to protect the security and privacy of data exchanged among IoT devices in a distributed untrusted IoT environment.

Recently, blockchain technology has gained popularity as a promising solution and become a research hot spot [6,11] due to its various salient features, such as security, anonymity, auditability, trust, transparency, and decentralization in the IoT domain [12–15]. IoT devices can rely on a blockchain to communicate and share data, since blockchain technology matches the distributed feature of an IoT network [16]. Authentication is ensured, since each transaction for each IoT node is signed by a private key, thus preventing man-in-the-middle attacks [17]. A blockchain is distributed in nature, thereby eliminating single points of failure. Furthermore, it offers immutability, meaning that a transaction cannot be changed once it is validated [18], so data manipulation can also be eliminated. However, direct application of existing blockchain technologies in the IoT domain is not possible, for example the core component of a blockchain is the consensus mechanism, which is responsible for ensuring consensus with the whole network before any transaction is added in the blockchain. The well-established consensus protocols used in blockchains are not suitable for IoT networks, since these protocols (e.g., PoW, PoS, and PoET) are computationally expensive, power hungry, low-throughput and subject to time delay [19]. Moreover, these consensus mechanisms were solely developed for cryptocurrency applications and lead to centralization issues since the node with highest mining power is selected to control a network [20]. Various recent studies have proposed different solutions to overcome these shortcomings. In [21], an IoT-oriented proof of honesty consensus mechanism is proposed for transaction validation which protects the scheme from participating in malicious or faulty nodes. In [7], a stochastic blockchain-based data integrity checking scheme for an IoT network is proposed which incorporates a data integrity checking process in the blockchain consensus mechanism. The authors in [12] proposed a fast scalable blockchain, called treechain, for IoT networks, which introduces randomization in the transaction level and blockchain level. A scalable blockchain titled Groupchain is proposed for fog-enabled IoT networks in [22]; also, to increase throughput and reduce confirmation latency, a reduced-size consensus mechanism is proposed. In [23] Spacechain, a 3-dimensional blockchain, is proposed, which overcomes the heterogeneity and scalability issues. Also, a new consensus mechanism is proposed to improve network security. In [24] a new lightweight consensus mechanism is proposed which does not include a complex mathematical algorithm to be solved by the mining node. This validates both trades and blocks. In [25] a lightweight consensus mechanism is proposed which eliminates any kind of extra resource consumption. In addition, a distributed trust management and throughput management strategy is introduced. Lastly, a dual vote confirmation-based consensus scheme is proposed for blockchain-enabled IoT networks in [26].

Though these newly proposed lightweight consensus algorithms improve performance in IoT networks, they do not consider data verification. This means that the consensus mechanisms only validate transactions by computing the hash value through a key sharing mechanism or voting method, but do not justify the originality of the data inside each transaction. Data could be falsified by an adversary, which could result in serious damage to the network in the final analysis output. A means of handling missing, erroneous or false data is urgently needed for proper operation of the industrial process. Artificial Intelligence approaches can be very efficient in several recognition [27–29], detection [30,31], and pattern recognition [32–34] tasks. AEs can be used to understand missing or corrupted data, since this type of neural network can model and learn the underlying patterns of the data. Hidden layers of the AE learn the true behavior of the input data. Therefore, if something unusual happens in the data generated by the IoT devices, the AE can easily predict unusual events in the input data.

In this article, we propose a permissioned Hyperledger Fabric blockchain-enabled IIoT network along with an AE-based consensus algorithm to verify the data before it is stored in the Blockchain. The contributions of this work are as follows.

- We first review the existing algorithms in blockchain-enabled IoT networks and summarize the consensus mechanisms along with their limitations in terms of IoT data originality verification.
- We propose a permissioned Hyperledger Fabric distributed ledger technology as the underlying blockchain architecture for the IIoT network instead of using any public blockchain network.
- To overcome the data originality verification problem identified in the recently proposed consensus mechanisms, we incorporate an AE with the Hyperledger CC in the consensus process for distinguishing anomalous data from original data. Therefore, using this AE-enabled CC-based consensus mechanism, the distributed ledger only stores original data.
- We have conducted a case study to illustrate the implementation of the above proposal, in which the AE is trained and tested on the IoTID20 dataset and MF is used to show the block architecture with only original data stored in the Fabric network.

In Section 2, recently proposed consensus algorithms in blockchain-based IoT networks are discussed along with their potential limitations and strengths. Section 3 describes the system model with Hyperledger Fabric architecture and the relevant operations of a Fabric-integrated IIoT network. In Section 4, we discuss the case study that demonstrates the implementation of the proposed scenario. Finally, Section 5 concludes this paper.

2. Related works

In this section, we discuss recently proposed blockchain-based solutions for IoT networks. We summarize the consensus mechanisms and their potential shortcomings in terms of data originality verification.

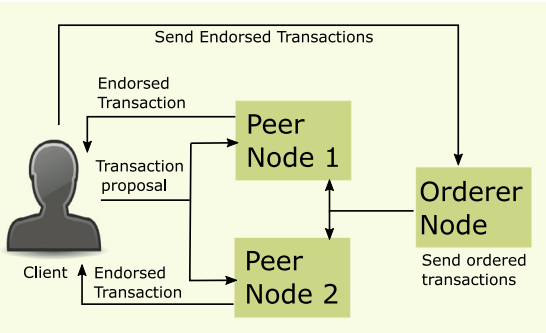


Fig. 1. Architecture of a Hyperledger fabric network, comprising a client node, endorsing peer node, committing peer node and master/origin node. The endorsing peer node verifies the transactions from the clients, the ordered node forms blocks of transactions and distributes them among the peer nodes, and the committing peer node validates the transactions based on the endorsement policy before storing each blocks.

Table 1
Comparisons of several consensus algorithms for blockchain enabled IoT networks.

Method	Strengths	Limitations
PLEDGE	Proof of honesty-based consensus mechanism; low latency, low computations, low communications complexity.	Data verification issue is not addressed in the time of transaction verification.
LDC	Lightweight data block structure; lightweight data consensus algorithm.	Consensus algorithm is ambiguous, Data verification is not clearly explained during the transaction verification process.
PoAh	Lightweight consensus mechanism; Highly scalable.	Only verifies block using DS and MAC address but do not verify the data which may result in verified block with false data.
PoBT	Block verification and trade validation based consensus mechanism	SC only verifies the permission of both end IoT device but does not verify data.
DVCC	Dual vote confirmation based consensus mechanism; Better throughput; Less delay .	However, only voting a block for validation does not ensure the data originality or elimination of malicious data.
Tree Chain	Fast; Optimized; Highly scalable; reduced overhead.	Validates blocks based on hash value but does not verify the data of the transactions.
3D-GHOST	Consensus process enhances the security.	However, this consensus mechanism lacks data verification.
Groupchain	Scalable for IoT network	Data verification of the transactions is not clearly shown.
LSB	Lightweight consensus mechanism; Distributed throughput management scheme; resilient against several security attacks.	The transactions are verified based on the hash and signature but no data verification is shown.

DS — Digital Signature, SC — Smart Contract.

PLEDGE [21] This method maintains an honesty metric for the nodes which propose blocks in the network. When a block is appended by a node in the network, the honesty metrics are updated. Cumulative honesty metrics are also computed for each node. A list of honest nodes is formed based on the value of each node's honesty metric. If a node's honesty metric is greater than a predefined threshold then it is included in the list. Primary and secondary block proposers are selected from the list at random to propose a block. If anything goes wrong, then the proposed block is rejected.

LDC [35] An Edge gateway node (EGN) works as a blockchain node. When an EGN receives a chunk of data, it generates the hash value of the data and includes it in the block of data which will be verified by another EGN. This data block is sent to the other EGN for verification purposes. The EGN which receives the data block checks whether or not the destination EGN and the hash value of the previous block of the data block to be verified are the same. If the two match, then the verification is true. On the other hand, if the destination EGN receives more than 50% of the verification result, then the data is marked correct. From this explanation, it is not clear how the data is verified. The proposed consensus algorithm only checks the hash value of the previous block and destination gateway, which does not explain the mechanism of verifying the originality of the data.

PoAh [36] This proposed proof of authentication consensus algorithm uses the Elgamal crypto system for encryption and decryption. The public key of each network node is made available to all other nodes for signature verification. Each network device first adds its digital signature before broadcasting a block in the network for verification. When a trusted node receives a block for verification, it first verifies the signature with the help of the source node's public key. During the second round of

verification, the trusted nodes verify the MAC address from the transaction. After successful block verification and authentication, the source node generates a hash of the block and adds it to the previous block in the chain. Other nodes follow the same process for adding a block to the chain. Moreover, a successful verification increases the trust value of a node and the trust value decreases if a fake block authentication occurs.

LSB [25] This consensus algorithm introduces randomness by generating blocks using overlay block managers (OBMs). Each OBM waits for a fixed amount of time before generating a new block. In the meantime, if another OBM generates a block, then an OBM which was waiting to generate a block will discard the transactions that are already included in other blocks generated by other OBMs. This avoids attempts at verification of duplicate transactions. The block generation time is also limited in such a way that each OBM can generate a new block at a given time interval. However, the generated blocks will be attached to the chain if and only if the transactions in the block are verified. The transactions are verified based on the hash value and signatures involving public and private keys.

PoBT [24] This proposed mechanism first verifies trades and then forms a consensus. An IoT device in the network generates trades and sends the trades as transactions to the source node (blockchain node) to which it is connected. This source blockchain node invokes a smart contract to verify the trade. The source node verifies that the destination IoT device can receive this trade according to the smart contract business logic. If the destination IoT device is allowed or involved in this trade exchange scheme, then a local consensus is executed. Otherwise, the destination blockchain node that is connected to the destination IoT device receives the trade. The destination blockchain node also invokes the smart contract and follows the above process specified in the smart contract. If this trading process for the two IoT devices is valid, the trade is sent to the ordered node for consensus formation. During this period of time, a number of session nodes are selected. These session nodes further verify the block of trades by keeping track of the source and destination nodes along with the trades with which they are involved.

DVCC [26] This consensus mechanism evaluates the credit score of each network node which will maintain the distributed ledger of the blockchain, conduct transaction verification, and perform other blockchain operations. To finalize a block that will be added to the distributed ledger as a valid block, more than 2/3 of the votes are required from both verifier nodes and member nodes. Two voting strategies are introduced. In the simple voting mechanism, each verifier node votes for the block and they have the same rights. On the contrary, in the weight-based voting mechanism, the voting right is made proportional to the credit score of each individual blockchain node.

Tree Chain [12] This consensus protocol introduces a randomization process for selecting a validator node which will commit or store transactions and blocks to the distributed ledger at low computational cost. This scheme uses the output of a hash function to commit transactions in the ledger. A validator node is selected randomly to commit a transaction depending on the most significant bit (MSB) of the hash of the transaction. The authors termed this MSB a "consensus code." In each transaction committal period, each validator node is assigned a particular consensus code, so a transaction is only committed in the ledger by randomly selecting a validating node.

3D-GHOST [23] The dynamic weight distribution is first considered when designing a 3D-GHOST consensus mechanism to better address the real-time distributed ledger state. Dynamic weights are formed based on three metrics, cardinal value, data validity and contact degree, where cardinal value is related to the block creation rate, data validity is related to valid transaction committal, and contact degree is related to the connection between the vertex and the entire ledger. In addition, a contact degree greedy traversal algorithm is used to compute the dynamic weights. Based on these two mechanisms (dynamic weights and greedy search algorithm), a 3D-GHOST consensus is developed which helps the peer nodes reach a consensus about the transactions in the network.

Groupchain [22]A number of blockchain nodes form a group in which one node becomes the leader node and the others act as member nodes. The leader node serializes all of the received transactions and creates a block. This newly generated block is broadcast to all member nodes in the group. When the member nodes receive this block for verification, the member nodes verify the transactions and broadcast the transaction results to each other along with their individual signatures. Upon receiving results from other member nodes, each member node matches the received results with its own verification results. If the block is considered a valid block, then the member nodes sign the block and forward it to the leader. If the majority of the results from the member nodes are valid, then the block is disseminated to all of the blockchain nodes for storage.

The summary of the above consensus mechanisms of blockchain-enabled IoT networks shows that all of the schemes are similarly complicated in terms of data verification. Some of the studies do not clarify the process of data verification. Table 1 shows the major contributions and limitations of these methods in terms of data verification.

3. System model

We have used Hyperledger Fabric as the underlying blockchain architecture. Fig. 1 shows the basic architecture of the Hyperledger Fabric network, in which a client sends transactions to the endorsing peer nodes. The endorsing peer nodes verify the transactions based on the CC (CC is the smart contract for Hyperledger Fabric) logic and return the endorsed transactions (signed with a private key) to the clients. The clients collect the endorsed transactions and forward these transactions to the orderer nodes. The orderer node creates a block of transactions and disseminates it to all of the peer nodes in the network. The committer nodes validate each transaction according to the endorsement policy and stores the validated transactions in the blockchain. In this work, we have integrated the Hyperledger Fabric blockchain platform into an industrial IoT network. The following subsections will illustrate the use of Hyperledger Fabric in the IoT network.

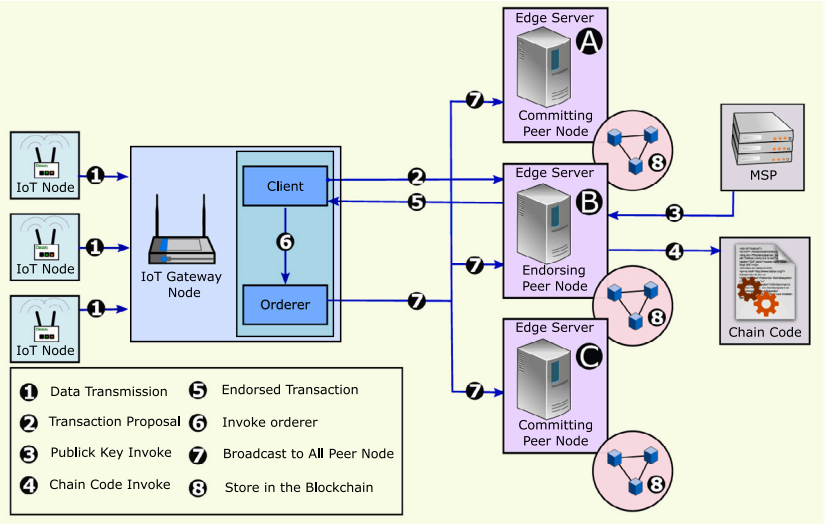


Fig. 2. Network architecture of Industrial IoT-integrated Hyperledger blockchain method. Each IoT device sends the sensed data to the IoT gateway node. The IoT gateway node performs two roles, as a client and an ordered node. The gateway node generates a transaction proposal, sends the transactions, collects the transactions, and orders the transactions. The edge servers work as both endorsing and committing nodes. The edge servers verify transactions through CC, reach a consensus, and store the block in the blockchain.

3.1. Network model

IoT devices are low-powered and do not have enough resources to support a blockchain ledger and other functionalities of Fabric. Therefore, the edge nodes will host the blockchain. The edge nodes are connected to each gateway node of the IoT devices. The IoT devices send the sensed data to the IoT gateway nodes. The gateway nodes forward these data as transactions to the edge nodes, where the consensus mechanism is conducted to verify the data and validate the transactions. Fig. 2 shows the overall architecture of the Fabric-enabled IoT network, in which each IoT device is connected to an immediate gateway node. The gateway node forwards the collected data to the edge nodes. Finally, the edge nodes conduct the consensus mechanism.

3.2. Network entities

3.2.1. IoT nodes

IoT nodes are low-powered sensor devices equipped with various types of sensors, such as a temperature sensor, humidity sensor, pressure sensor, and so on. Each IoT device also consists of a microprocessor, a power unit and a communication transceiver. In an industrial scenario, the IoT devices are attached to the industrial machinery in order to measure the performance of that machinery. An IoT device senses the intended data and forwards that data to the IoT gateway node.

3.2.2. IoT gateway node (as client node)

A client represents an application that generates a transaction proposal. In a business blockchain network, a client node works on behalf of an end user. However, in the IIoT scenario, each IoT gateway node operates as a client node. The IoT gateway node communicates with a peer node of the blockchain network to exchange data as payload over the transaction proposal message. This node is responsible for transaction generation and endorsed transaction reception.

3.2.3. Edge server (as peer node)

A peer node can be defined as a blockchain node that keeps a copy of the distributed ledger and stores all transactions as blocks in the ledger [37]. Since this type of node requires sufficient resources, such as memory and computing power, we have assigned the edge server to this role. Some edge servers as peer nodes will also work as endorsing nodes. As an endorsing node, an edge server executes the transactions from the client node following the logic implemented in the CC and endorses the transactions. Moreover, a group of endorsing edge server nodes verifies the transactions by invoking the CC and reaches a consensus in the network.

3.2.4. IoT gateway node (as ordered node)

Since a node is a logical entity and multiple nodes can reside in the same physical device [38], therefore we also considered an IoT gateway node as an orderer node. As an orderer, the IoT gateway node orders the received transactions in ascending or descending order and creates a block of transactions. This block of transactions is broadcasted to all peer nodes in the blockchain network to store the transactions in the ledger.

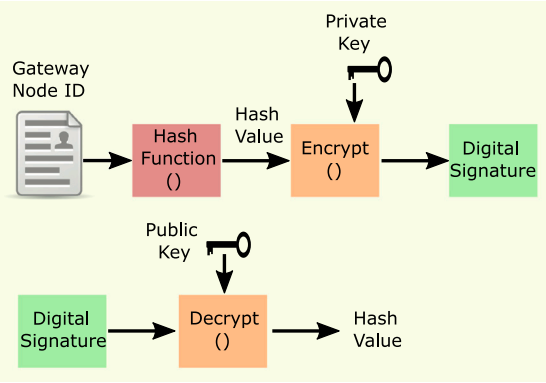


Fig. 3. Identity verification procedure by the peer node. Each peer node first decrypts the digital signature using the corresponding public key and extracts the hash value. Then, it computes the hash value of GWN_{ID} from the proposal message.

3.2.5. Edge server (as committer node)

A committer node in a Hyperledger Fabric blockchain system only contains a copy of the distributed ledger. In our scenario, an edge server also works as a committer node. All edge servers contain a copy of the ledger. However, some edge servers host both the chain code and a ledger copy, so they work as both endorsing nodes and committer nodes. As a committer node, an edge server stores the block of the transactions in the ledger following the endorsement policy.

3.2.6. Membership service provider

Membership Service Provider (MSP) is a Fabric element that handles certificates, key management, authentication and other cryptography protocols. In our IoT network, the MSP will store the public key of each of the IoT Gateway nodes. MSP will provide the public keys to the CC during identity verification process.

3.3. Transaction propagation

3.3.1. Transaction proposal

After obtaining data from the IoT nodes, an IoT gateway node generates a payload. These payloads will be attached to the transaction proposal message along with other identifiers of the transactions (TX). A transaction proposal message can be formed as below

$$TX = \{GWN_{ID}, TX_{ID}, CC_{ID}, Payload, GWN_{Sign}\}$$

Where, GWN_{ID} is the unique identifier of the IoT Gateway node, Tx_{ID} is the ID of the transaction, CC_{ID} is the ID of the specific CC to be invoked, Payload contains the data of the IoT nodes and GWN_{Sign} denotes the digital signature of the GWN_{ID}. Payload itself can be defined as,

$$Payload = \{IoTnodeID, Data, timestamp\}$$

After forming the transaction proposal message, the GWN sends this proposal message to the endorsing peer nodes of the Hyperledger Fabric network. The GWN also keeps a copy of the transaction in its memory. In Fig. 2, step 1 shows the IoT nodes sending data to the IoT GWN, and step 2 shows the IoT GWN sending the transaction proposal message to an endorsing peer node such as an edge server (B).

3.3.2. Transaction endorsement

Upon receiving the transaction proposal message from the GWN, the peer node (endorsing peer) (B) in Fig. 2 executes the transaction, meaning that it verifies the transaction based on the verification logic written in the CC by invoking the appropriate CC_{ID} as mentioned in the transaction proposal message. CC invocation is shown in step 4 of Fig. 2. Before verifying the payload, the endorsing peer will check the ID of the IoT node by invoking the public key from MSP. After executing the transaction through the CC logic, the endorsing peer endorses the transaction by signing it and then returns it to the GWN. We discussed the CC operations in detail in Section 3.4. The below mathematical notation denotes the return message from the endorsing peer to the GWN.

$$R_{TX} = \{TX_{ID}, TX - Verified, Tx - Endorsed, EP_{sign}\}$$

or similarly,

$$R_{TX} = \{Tx_{ID}, Tx - NotVerified, Tx - Endorsed, EP_{sign}\}$$

Where, TX_{ID} denotes the transaction that is executed and endorsed, $TX-Verified$ means the payload in the transaction passed the verification logic, $TX-Endorsed$ means the transaction is executed by the Endorsing peer and finally, the EP_{sign} denotes the sign of the Endorsing peer which executed the TX.

3.3.3. Distributing blocks

As shown in Fig. 2, step 6, the client node sends all collected endorsed transactions to the orderer node. The orderer node first orders all the transactions in ascending or descending order, and then it creates a block. Next, it calls the Broadcst() function to disseminate the newly created block to all edge servers (peer nodes) in the network, as shown in step 7 of Fig. 2.

3.3.4. Block committing

Upon receiving the ordered transactions of the block, each peer node (peer nodes (A), (B), (C)) in Fig. 2 iterates over each transaction and validates it according to the endorsement policy set by the network administrator. Any transactions that are not verified are discarded from the block. Only verified and validated endorsed transactions are stored in the blockchain by invoking the commit() function.

3.4. Consensus mechanism & transaction verification

Since there is no central authority in the blockchain realm, any decisions in a blockchain network can be made through a consensus protocol. Consensus refers to a method of reaching an agreement in a group such that the agreement benefits the entire group. Consensus decision making is a group decision making process in which the group members develop and agree to support a decision that is in the group's best interest on the whole. The most popular consensus mechanisms in blockchain networks are PoW, PoS, and PoAh. Use of these consensus mechanisms would introduce complexity in resource-constrained IoT networks, and IoT-suitable consensus mechanisms do not address the verification of the originality of the IoT data, as discussed in Section 2. Therefore, here, we have integrated an unsupervised machine learning algorithm with CC to reach a consensus and to verify the data contained in the transactions from the end IoT devices. The committing nodes then generate a hash value and append it to the block before adding the verified block to the blockchain. The AE is used as the underlying data verification machine learning architecture. The AE takes the data generated by the IoT devices as input and tries to predict any abnormalities in the data according to the training data or learned behavior. Fig. 4 shows the consensus process, in which each endorsing node invokes the CC to verify the transactions. The CC itself verifies the ID and data of the payload contained in the transaction, as described below. If 2 out of 3 endorsing peer nodes agree on the verified result, then the data is considered valid and original, and thus suitable for storage in the ledger. 2 out of 3 verification result is considered as the endorsement policy.

To verify a transaction generated by an IoT GWN, each peer node invokes a CC according to the CC_{ID} mentioned in the transaction proposal message. The two portions of the transaction are verified sequentially.

3.4.1. Identity verification

In the first phase, the ID of the IoT GWN is verified. The endorsing peer node first extracts the GWN ID and GWN digital signature from the transaction proposal message. Then, the endorsing node invokes the public key of the GWN from the MSP and gets a hash value by decrypting the digital signature. At the same time, the endorsing node also generates a hash value from the GWN_{ID}. If the two hash values are the same, then the GWN_{ID} is verified. This operation is illustrated in Fig. 3.

3.4.2. Data verification

The AE neural network comprises an encoder and a decoder, which are the neural network with weights and biases. The encoder part of the AE compresses the input data of the transaction's payload into a low-dimensional representation. On the contrary, the decoder tries to regenerate the input data based on the low-dimensional output of the encoder. The following mathematical equations show the working mechanism of an AE [39].

- Input layer: A n dimensional vector V represents the input values for the encoder or input layer. The vector V can hold different kinds of data for instance, temperature readings from the sensors, electrical data, pressure data etc. Mathematically, the vector V can be written as $V = \{v_1, v_2, v_3, \ldots, v_n\}$ where the v_n denotes data. The encoder uses the mapping function in Eq. (1) to generate low-dimensional data from the input or training data.

$$h_i = f_\theta(x_i) = s(Wx_i + b) \tag{1}$$

  Where, x_i is the training dataset given as input to the AE. $W = d' \times d$, d' denotes the hidden layer of the encoder neural network. b denotes the bias value and finally s defines the activation function.
- Hidden layer: The purpose of the hidden layer of the AE is to learn the pattern or behavior of the dataset from the input layer so that it can identify any misbehavior or anomaly in unseen data. AE can multiple hidden layer but to reduce computational overhead, one single layer is used and one hidden layer performs well in practice [40].

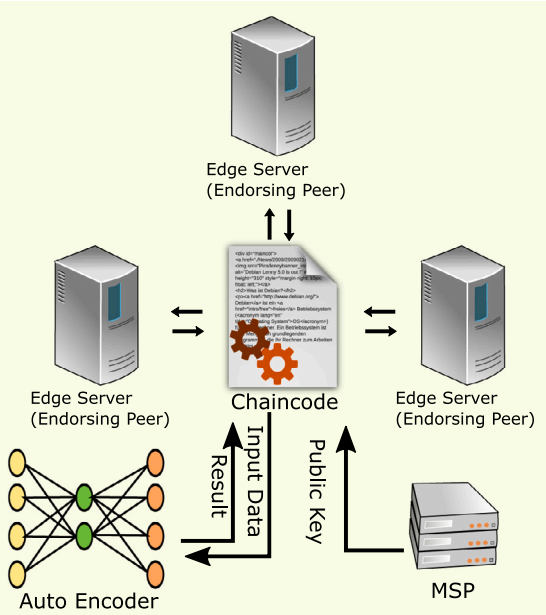


Fig. 4. The consensus process of the proposed Hyperledger-based IoT network. Three endorsing peer nodes invoke the chaincode to verify the ID and data from the payload. The votes of at least 2 out of 3 peer endorsing nodes are required to complete the consensus process.

- Output layer: In case of output layer, the generated output from this layer has the same dimension as input layer and mathematically it can be represented as $V' = v'_1, v'_2, v'_3, \ldots, v'_n\}$. decoder regenerates the input from the low dimensional output of the encoder through the mapping function as stated below-

$$h_i = g_{\theta'}(x_i) = s(W'h_i + b') \tag{2}$$

Similarly, W' is the weight matrix and b' is the bias value.

To learn the hidden pattern of the input data, MSE can be used as a loss function in neural networks. When a trained AE is tested on a dataset that is similar to the training dataset, the AE will provide low reconstruction error, compared to high reconstruction error for anomalous test dataset. To distinguish among anomalous IoT data and normal IoT data, a threshold value should be defined based on the reconstruction error. If the reconstruction error is higher than the threshold value for a new, unseen dataset then the new dataset is considered anomalous data.

4. Case study

We have conducted a case study to implement our proposed method. This section describes that case study that is divided into two subsections. The first subsection shows the training, testing and performance metrics of the AE model for the IoTID20 dataset. The second subsection shows the CC deployment and working principles of CC with the corresponding block architecture of Fabric.

4.1. Autoencoder

4.1.1. Data pre-processing

For the case study, we have considered the IoTID20 dataset [41]. This dataset is comprised of 80 features of IoT network and has 3 labels. These labels include binary (normal and anomaly), categorical (normal, DoS, Mirai, MITM, Scan) and sub categorical (normal, Syn flooding, Brute force, HTTP flooding, UDP flooding, ARP spoofing, Host Port, OS). Since our proposed model only considers the separation of anomalous data from original data, we discarded the categorical and sub-categorical labels from the IoTID20 dataset. The dataset contains 40,073 instances of normal and 585,710 instances of anomalous examples. We replaced the infinite and missing values with a Python numpy-based empty placeholder (nan). Also, to scale/normalize the features of the dataset between 0 and 1, we applied the normalized function. Since variables or features measured in various scales may cause bias during model fitting, the Python based open-sourced *scikit-learn* library's *MinMaxScalar* function was used to scale all variables in the range of 0 to 1.

The dataset was split into training and testing sets (67% and 33%, respectively). That is, 67% of the shuffled normal and anomalous dataset was used to train the AE model and 33% of the dataset was used to test the model. The number of input, hidden and output layer units of the AE is 77,8 and 77, respectively. The model was trained over 100 iterations (epochs) with a batch size of 100. Fig. 5 shows the training loss and validation loss curves. The two are similar to each other and training the model for more than 100 epochs merely stabilizes the losses. Based on the loss, we set the threshold for anomaly detection at 0.15.

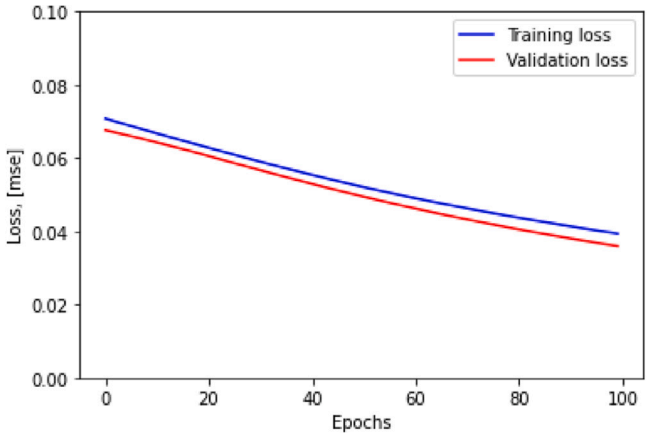


Fig. 5. This figure depicts the learning curve of the AE model. It shows how loss decreases with the increasing epochs.

Table 2
Detection results of test data.

Loss_mae	Threshold	Anomaly
0.027934	0.15	False
0.029030	0.15	False
0.038291	0.15	False
0.197861	0.15	True
0.056262	0.15	False
0.172202	0.15	True
0.172211	0.15	True
0.130087	0.15	False
0.028450	0.15	False
0.191043	0.15	True

4.1.2. Evaluation

To evaluate the trained AE with the IoTID20 dataset, we used accuracy, precision and recall metrics. These metrics were computed using the following equations based on [42]

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

This equation gives the accuracy of the model, or the portion of the total dataset that is classified correctly by the AE.

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

Precision defines the ratio of the total number of correctly classified anomalous data points to the total number of predicted anomalous data points.

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

Recall is also known as sensitivity. This metric defines the ratio of the total number of correctly classified anomalous data points to the total number of anomalous data points.

In the above equations, TP, TN, FP, and FN represent the true positive rate, true negative rate, false positive rate and false negative rate, respectively. The TP, TN, FP and FN values, which were 185,500, 12, 13,290, and 7,585, respectively, were used to form a confusion matrix. Fig. 6 shows the evaluation results. We obtained an accuracy of 89.89%, precision of 93.31% and recall of 96%. The high precision and recall values indicate good performance. Also, Table 2 shows the detection results. It is clear that, when the loss value for the test dataset is higher than the defined threshold, the data is classified as anomalous.

4.2. Chaincode deployment

We have used MF to show the implementation of our CC inside fabrics endorsing peer node and to show the working of our CC. MF is a Hyperledger fabric network tool developed by Tong Li which enables one to setup and expand fabric network, deploy and install CC, invoke transactions, storing the transactions in the blocks and configuring the channels [43]. CC represents a piece of program which can be written by GO, Java or NodeJS. We have used Java for implementing our CC inside MF framework of Hyperledger fabric. In our implementation, we have used the methods provided by the CC stub interface that is *init()* and *invoke()*

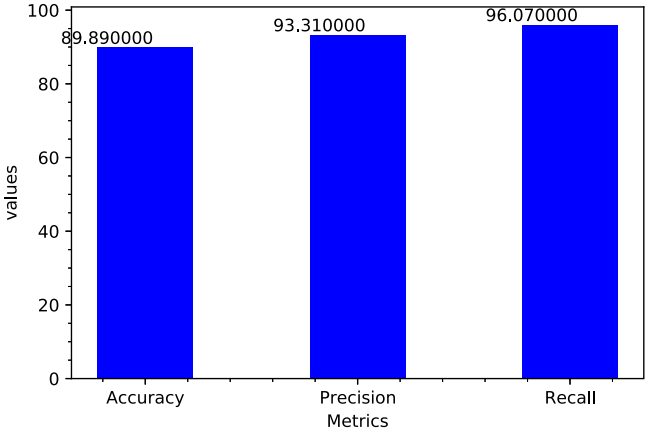


Fig. 6. Evaluation results of the trained AE model based on accuracy, precision and recall.

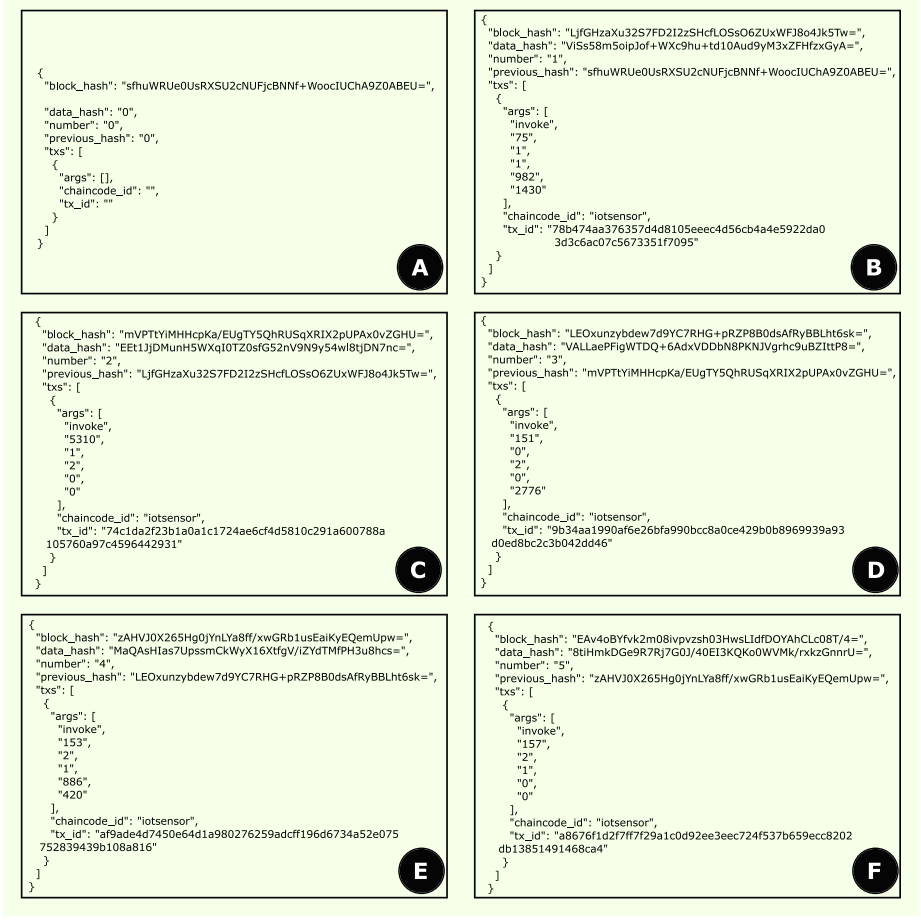


Fig. 7. Stored transactions in the blocks of the blockchain. After verification through the AE, the data of the IoT nodes will be stored in the blockchain. (A) block is the genesis block and the remaining blocks represent 5 different blocks with different data with transaction ID.

methods. The *Init*() method is responsible for initializing the parameters when the CC is installed at each endorsing peer node. On the contrary, the *invoke()* method is called when a new transaction is received by the endorsing peer node. This *invoke()* method conducts read and write operations in the ledger. In MF we have put our CC Java code inside the CC directory of the vars directory of Minifabric (Minifabric creates a sub directory inside the working directory where all the required scripts and files are generated

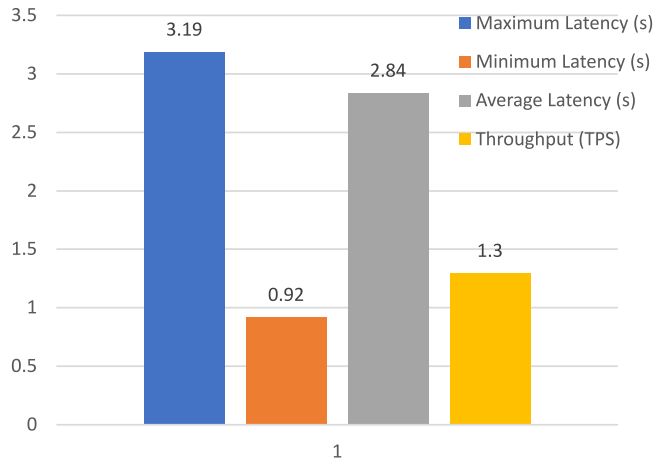


Fig. 8. Performance results of the chaincode within the blockchain framework. The performance metrics are: maximum latency (3.19 s), minimum latency (0.92 s), average latency (2.84 s) and throughput (1.3 transactions per second).

through corresponding commands). First a fabric network is started through *minifab up* command which additionally creates a channel and connects all the nodes through this channel. By default a built in CC installs and instantiates in the nodes. Therefore we installed our own CC at each peer node (docker container) through the CC installation command *minifab install -n iotsensor -v 1.0*. After installation, the CC is approved, committed and initialized by the corresponding Minifabric commands. After successful installation of the CC, we have called the *invoke()* method through the invocation command for passing some sample data as payload of a transaction. The *invoke()* method itself calls the trained AE model for verifying the data. The AE differentiates the anomalous data from the normal data and returns the result as Boolean value to the CC. Based on the results the CC writes those sample data inside blocks of the fabric (if the returned result is false meaning the passed data is not anomalous). Fig. 7 shows the output of the block query command of MF where we have called the *invoke()* method 5 times and passed different data as payload. Fig. 7 (A) shows the genesis block structure and the rest of them illustrates the generated block with the verified data from the transactions. We have also tested our chaincode using Hyperledger caliper module integrated in MF through *minifab caliperrun* command. Fig. 8 shows the performance from caliper report in terms of latency and throughput. It shows that the maximum latency is 3.19s and the throughput is 1.3 TPS.

5. Conclusion

Data originality verification is essential in data-driven industrial IoT networks to ensure proper maintenance and control of industrial machinery. False data may be generated due to faulty IoT nodes or anomalous data may be injected by nefarious actors who provide misleading information to the application layer. Recently proposed consensus mechanisms in blockchain-enabled IIoT networks do not consider data verification issues, and may thus allow storage of anomalous data in the ledger. Also, several previous studies used public blockchain networks for IIoT networks, which are less secure then private blockchain technologies. To overcome these problems, we proposed a Hyperledger Fabric network, which is a permissioned blockchain network, and incorporated a network anomaly detection machine learning model, AE, with the CC to reach a consensus in the network. The CC invokes the AE after receiving a transaction from the client nodes. Only the verified original data from the AE is stored in the blockchain through the CC. We trained and tested the AE using the IoTID20 dataset and used Minifabric to illustrate the CC operation and block structures with verified data. In our future work, we will reduce the complexity of chaincode and enhance the performance in terms of throughput and latency. Also, the compatibility of using AE in Blockchain IoT system will be investigated in our future work.

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.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP), South Korea grant funded by the Korea government (MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub).

References

[1] Y. Lu, L. Da Xu, Internet of Things (IoT) cybersecurity research: A review of current research topics, IEEE Internet Things J. 6 (2) (2018) 2103–2115.
[2] J.H. Kim, 6G and Internet of Things: a survey, J. Manag. Analyt. 8 (2) (2021) 316–332.
[3] L. Da Xu, W. He, S. Li, Internet of Things in industries: A survey, IEEE Trans. Ind. Inf. 10 (4) (2014) 2233–2243.
[4] M. Younan, E.H. Houssein, M. Elhoseny, A.A. Ali, Challenges and recommended technologies for the industrial internet of things: A comprehensive review, Measurement 151 (2020) 107198.
[5] N. Integra, 7 Industrial IoT applications, Nexus Integra (EN), 2020, URL https://nexusintegra.io/7-industrial-iot-applications/.
[6] C. Zhang, Y. Chen, A review of research relevant to the emerging industry trends: Industry 4.0, IoT, blockchain, and business analytics, J. Ind. Integr. Manag. 5 (01) (2020) 165–180.
[7] Y.-J. Chen, L.-C. Wang, S. Wang, Stochastic blockchain for IoT data integrity, IEEE Trans. Netw. Sci. Eng. (2018).
[8] S. Hameed, F.I. Khan, B. Hameed, Understanding security requirements and challenges in Internet of Things (IoT): A review, J. Comput. Netw. Commun. 2019 (2019).
[9] D. Diaz Lopez, M. Blanco Uribe, C. Santiago Cely, A. Vega Torres, N. Moreno Guataquira, S. Moron Castro, P. Nespoli, F. Gomez Marmol, Shielding IoT against cyber-attacks: An event-based approach using SIEM, Wirel. Commun. Mob. Comput. 2018 (2018).
[10] J. Qu, Blockchain in medical informatics, J. Ind. Inf. Integ. 25 (2022) 100258.
[11] S. Khan, R. Singh, Kirti, Critical factors for blockchain technology implementation: A supply chain perspective, J. Ind. Integ. Manag. (2021) 2150011.
[12] A. Dorri, R. Jurdak, Tree-chain: A fast lightweight consensus algorithm for IoT applications, 2020, arXiv preprint arXiv:2005.09443.
[13] M.M. Arifeen, A. Al Mamun, M.S. Kaiser, M. Mahmud, Blockchain-enable contact tracing for preserving user privacy during COVID-19 outbreak, 2020, Preprints.
[14] M.M. Arifeen, A. Al Mamun, T. Ahmed, M.S. Kaiser, M. Mahmud, A blockchain-based scheme for Sybil attack detection in underwater wireless sensor networks, in: Proceedings of International Conference on Trends in Computational and Cognitive Engineering, Springer, 2021, pp. 467–476.
[15] H.M. Hussien, S.M. Yasin, N.I. Udzir, M.I.H. Ninggal, S. Salman, Blockchain technology in the healthcare industry: Trends and opportunities, J. Ind. Inf. Integr. 22 (2021) 100217.
[16] A. Gorkhali, L. Li, A. Shrestha, Blockchain: A literature review, J. Manag. Analyt. 7 (3) (2020) 321–343.
[17] S.S. Panda, U. Satapathy, B.K. Mohanta, D. Jena, D. Gountia, A blockchain based decentralized authentication framework for resource constrained IOT devices, in: 2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT, IEEE, 2019, pp. 1–6.
[18] E. Politou, F. Casino, E. Alepis, C. Patsakis, Blockchain mutability: Challenges and proposed solutions, IEEE Trans. Emerg. Top. Comput. (2019).
[19] D. Puthal, S.P. Mohanty, Proof of authentication: IoT-friendly blockchains, IEEE Potent. 38 (1) (2019) 26–29, http://dx.doi.org/10.1109/MPOT.2018.2850541.
[20] A. Li, X. Wei, Z. He, Robust proof of stake: A new consensus protocol for sustainable blockchain systems, Sustainability 12 (7) (2020) 2824.
[21] I. Makhdoom, F. Tofigh, I. Zhou, M. Abolhasan, J. Lipman, PLEDGE: A proof-of-honesty based consensus protocol for blockchain-based IoT systems, in: 2020 IEEE International Conference on Blockchain and Cryptocurrency, ICBC, IEEE, 2020, pp. 1–3.
[22] K. Lei, M. Du, J. Huang, T. Jin, Groupchain: Towards a scalable public blockchain in Fog computing of IoT services computing, IEEE Trans. Serv. Comput. 13 (2) (2020) 252–262.
[23] M. Du, K. Wang, Y. Liu, K. Qian, Y. Sun, W. Xu, S. Guo, Spacechain: A three-dimensional blockchain architecture for IoT security, IEEE Wirel. Commun. 27 (3) (2020) 38–45.
[24] S. Biswas, K. Sharif, F. Li, S. Maharjan, S.P. Mohanty, Y. Wang, PoBT: A lightweight consensus algorithm for scalable IoT business blockchain, IEEE Internet Things J. 7 (3) (2019) 2343–2355.
[25] A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, LSB: A lightweight scalable blockchain for IoT security and anonymity, J. Parallel Distrib. Comput. 134 (2019) 180–197.
[26] Z. Qiu, J. Hao, Y. Guo, Y. Zhang, Dual vote confirmation based consensus design for blockchain integrated IoT, in: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2020, pp. 1–7.
[27] T. Ghosh, S.M. Chowdhury, M.A. Yousuf, et al., A comprehensive review on recognition techniques for bangla handwritten characters, in: 2019 International Conference on Bangla Speech and Language Processing, ICBSLP, IEEE, 2019, pp. 1–6.
[28] M.H. Al Banna, T. Ghosh, M.J. Al Nahian, K.A. Taher, M.S. Kaiser, M. Mahmud, M.S. Hossain, K. Andersson, Attention-based Bi-directional long-short term memory network for earthquake prediction, IEEE Access 9 (2021) 56589–56603.
[29] T. Ghosh, M.M.-H.-Z. Abedin, S.M. Chowdhury, Z. Tasnim, T. Karim, S.S. Reza, S. Saika, M.A. Yousuf, Bangla handwritten character recognition using MobileNet V1 architecture, Bull. Electr. Eng. Inf. 9 (6) (2020) 2547–2554.
[30] M.H. Al Banna, T. Ghosh, K.A. Taher, M.S. Kaiser, M. Mahmud, A monitoring system for patients of autism spectrum disorder using artificial intelligence, in: International Conference on Brain Informatics, Springer, 2020, pp. 251–262.
[31] T. Ghosh, H. Al Banna, N. Mumenin, M.A. Yousuf, et al., Performance analysis of state of the art convolutional neural network architectures in Bangla handwritten character recognition, Pattern Recognit. Image Anal. 31 (1) (2021) 60–71.
[32] M.J. Al Nahian, T. Ghosh, M.N. Uddin, M.M. Islam, M. Mahmud, M.S. Kaiser, Towards artificial intelligence driven emotion aware fall monitoring framework suitable for elderly people with neurological disorder, in: International Conference on Brain Informatics, Springer, 2020, pp. 275–286.
[33] M.J. Al Nahian, T. Ghosh, M.H. Al Banna, M.A. Aseeri, M.N. Uddin, M.R. Ahmed, M. Mahmud, M.S. Kaiser, Towards an accelerometer-based elderly fall detection system using cross-disciplinary time series features, IEEE Access 9 (2021) 39413–39431.
[34] T. Ghosh, M.H. Al Banna, M.J. Al Nahian, K.A. Taher, M.S. Kaiser, M. Mahmud, A hybrid deep learning model to predict the impact of COVID-19 on mental health form social media big data, 2021, Preprints.
[35] W. Zhang, Z. Wu, G. Han, Y. Feng, L. Shu, LDC: A lightweight dada consensus algorithm based on the blockchain for the industrial Internet of Things for smart city applications, Future Gener. Comput. Syst. (2020).
[36] D. Puthal, S.P. Mohanty, V.P. Yanambaka, E. Kougianos, Poah: A novel consensus algorithm for fast scalable private blockchain for large-scale iot frameworks, 2020, arXiv preprint arXiv:2001.07297.
[37] H. Sukhwani, Performance modeling & analysis of hyperledger fabric (permissioned blockchain network), (Ph.D. thesis), 2019.
[38] Architecture Explained, Hyperledger, URL https://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html.
[39] F. Farahnakian, J. Heikkonen, A deep auto-encoder based approach for intrusion detection system, in: 2018 20th International Conference on Advanced Communication Technology, ICACT, IEEE, 2018, pp. 178–183.
[40] T. Luo, S.G. Nagarajan, Distributed anomaly detection using autoencoder neural networks in wsn for iot, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–6.
[41] I. Ullah, Q.H. Mahmoud, A scheme for generating a dataset for anomalous activity detection in IoT networks, in: Canadian Conference on Artificial Intelligence, Springer, 2020, pp. 508–520.
[42] R.C. Aygun, A.G. Yavuz, Network anomaly detection with stochastically improved autoencoder based models, in: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing, CSCloud, IEEE, 2017, pp. 193–198.
[43] Minifabric: A Hyperledger Fabric Quick Start Tool (with Video Guides) – Hyperledger, URL https://www.hyperledger.org/blog/2020/04/29/minifabric-a-hyperledger-fabric-quick-start-tool-with-video-guides.