International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 10, 2020
ISSN: 1475-7192

Decentralized Security Architecture Based On Software Defined Networking (Sdn) In Blockchain For Iot Network

[1]Anand Deepak George Donald

Abstract
Cloud computing has emerged as a major technology for delivering infrastructure and data service needs at cheap cost, with minimum effort and great scalability, and has therefore been widely adopted in the IT sector. Although there has been a tremendous increase in Cloud Computing use, information security issues have yet to be entirely addressed. To address the current challenges, this paper proposes a decentralised security architecture for the IoT network in the smart city based on Software Defined Networking (SDN) combined with blockchain technology that relies on the three core technologies of SDN, Blockchain, and Fog as well as mobile edge computing to detect attacks in the IoT network more effectively. Our findings show that the suggested decentralised security architecture outperforms centralised and distributed security architectures in the IoT ecosystem and takes less time to prevent threats. Our results also show that the architecture might be used with the IoT ecosystem as a security detection component that monitors and analyses the whole IoT ecosystem's traffic data to identify and prevent possible threats.

Keywords: *Security, Architecture, Blockchain, IoT, Network*

INTRODUCTION

Blockchain Technology

Blockchain technology is the way of the future for companies seeking to better security and privacy. Without a central authority, blockchain is a distributed ledger that maintains tamper-evident data in the form of a chain. Nodes are the participants or devices in the blockchain technology. The blockchain technology creates a decentralised network in which all network nodes actively participate in validating and verifying data. Cryptography will be used to encrypt the data that will be stored on the blockchain. Every block has an encrypted hash, a timestamp, and the hash of the preceding block in the chain to which it will link. As a result, the data on the blockchain is tamper-proof. The data is secured using blockchain, and people who participate in the network will be vetted, removing the data's privacy worry..[1]

Characteristics of Blockchain

Requirements define all of the qualities and attributes that a Blockchain system should have, as well as the constraints that the system must function within to be successful and efficient. As a result, they have an impact on the Blockchain system's general operation and may have a bigger impact on the system's architecture. The following are some of the blockchain network's non-functional characteristics.

- Openness: Due to the compatible nature of the nodes, blockchain has the capacity to utilise and exchange information throughout a transaction.
- Concurrency: As more nodes process data at the same time, blockchain speed increases.
- Scalability: Blockchain is scalable because it allows for the addition and deletion of additional nodes.
- When it comes to scalability, the three parameters that are most important are:
- Size: delay or manageability: Transaction Processing Rate of Distribution:
- Fault tolerance: Because the Blockchain network is fault-tolerant, a failure at any node will be invisible to all other nodes in the Blockchain, allowing the network to function normally even if there is a fault.
- Transparency: All nodes in the Blockchain network can see each other's transactions.
- Data security: The Blockchain network uses sophisticated cryptographic methods like SHA-255 to safeguard data.
- Quality of Service (QoS) on the Blockchain network is determined by response time and dependability, as well as the time it takes for a transaction to complete and the commitment to offer the essential services.
- Failure Management: A process must be in place to ensure that the Blockchain network is stable and to identify the root cause of failure. It may automatically recommend ways to recover from failure. [6]

[1] LNCT University Bhopal

Related surveys

Some surveys on trust mechanisms in cloud computing settings have previously been conducted. A. Horvath III et al. investigated customer trust in cloud computing systems in order to assist service providers in improving their practises. S. Harbajanka and P. Saxena reviewed trust techniques in cloud computing, highlighting the benefits and drawbacks of related research. E. Rawashdeh et al. provided a thorough overview of existing cloud trust models. J. Huang and D. Nicol conducted a study of current trust mechanisms and identified their shortcomings. T. Noor et al. gave an overview of cloud service trust management and identified outstanding concerns. M. Monir et al. presented a survey of cloud computing trust solutions for evaluating service providers' performance. M. Chandni et al. presented an overview of current trust-based strategies before discussing probable assaults on cloud systems. J. Lansing and A. Sunyaev created a conceptual model to define trust in the cloud and surveyed 43 relevant techniques..[7]

Cloud Computing

There are millions of websites hosted on the web in modern Internet age. The hosted site requires a large number of servers, which is highly expensive. Those servers' traffic rates must be steady, and they must be regularly checked and maintained. There will be a need to recruit additional staff to organise and maintain these servers. All of the data will be stored in data centres. As a result, continual attempts to maintain the server problem, as well as the workers, may detract from our ability to meet our business objectives. We are using "Cloud Computing" to prevent this time-consuming upkeep. "Cloud computing is the technique of storing, managing, and processing data from anywhere in the globe utilising a network of distant servers. It replaces a local server or a personal computer". Cloud computing services, such as data storage and application delivery, are given to the devices of the company over the internet. Cloud computing offers a number of benefits by merging data centres, resources, and servers across the internet..[1]

QR Code

QR codes are an effective information transmission medium that are utilised in a variety of applications today, including product traceability, mobile payment, advertising, degree and transcript creation, passport verification, and many more. When attacked with defacement, QR codes are divided into 40 symbol variants (to carry multiple data payloads) and four user-selectable error correction levels (ECL): L, M, Q, and H, which may correct up to 7%, 15%, 25%, and 30% error, respectively. The Reed-Solomon error correction method is used by QR codes to identify fault tolerance. The error correction codewords are formed using the Reed-Solomon algorithm and then appended to the tail of the QR code data codewords. The greater the QR code version and the error correction level, the more data payload and data retrieval reliability it may provide..

Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique in which an intelligent agent, a computer system that senses its environment, takes autonomous behaviours in order to receive cumulative rewards based on the information acquired from the environment. An intelligent agent is shown in Figure 3 as a typical diagram. Optimizing performance, power consumption, or thermal efficiency, or a combination of these, might be the reward here.
DELM computing technology may be used to make blockchain-based apps smarter. The distributed ledger's security may be increased by employing DELM. DELM may also be used to shorten the time it takes to grasp something by improving information exchange pathways. It also offers the opportunity to develop frameworks by employing the centralised architecture of blockchain technology. The smart application here collects data from a variety of sources, including cameras, smart devices, and IoT systems. As a component of smart applications, data from such approaches was assessed. These smart apps use blockchain as a fundamental component. Nonetheless, the DELM framework may be used to understand (data analysis and real-time analysis) and anticipate such application data. Data sets utilised by DELM models are processed over a blockchain network..[5]

OBJECTIVES OF THE STUDY

1. To study on Characteristics of Blockchain
2. To study on **Blockchain Technology**

RESEARCH METHOD
Network Self-Clustering

The clustering strategy for the IoT network is shown in Figure 1. The suggested network clustering algorithm's flow is shown in Figure 6. The grouping is done with the use of metaheuristic algorithms, as can be observed. The foundation of metaheuristic algorithms is a strong relationship between computational processes and optimization. The fundamental

benefit of these approaches is that they are free of local optimum spots. As a result, these methods scan the whole search space..

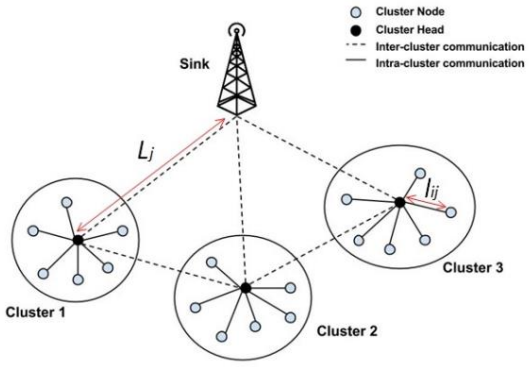

Figure 1. Network clustering scheme for cellular IoT network.

GA is a subset of metaheuristic approaches that uses a population-based methodology. For the search of a problem space, it displays strong global-based exploration performance [59]. As a result, GA is presented in this study for clustering heterogeneous IoT networks. In order to assess a single solution, an effective local-based exploration mechanism inside the search space is also necessary. While SA shows excellent results in this regard, this article opts for a hybrid technique (based on GA and SA) to improve the suggested IoT network clustering.

Optimization of Distance and Coverage by GA

Because of its self-organized characteristic, the GA technique aids in distance optimization. As shown in Figure 2, the following GA begins using the previous step GA's final global solution as the beginning solution. This phase effectively expresses the mobility of various nodes. The suggested GA approach additionally takes into consideration network coverage optimization. The second GA phase adds a local search technique to the previous phase's GA solution. The overall network energy dissipation is minimised by optimising the distance between CH and a node, as well as the distance between CH and the sink or edge computing node. The distance-based equation is used in the previous stage to cluster the nodes into several groups and specify the number of clusters using the GA algorithm. The best answer from the previous phase is used to produce the beginning population for the current stage. The GA optimization stage employs a multi-objective cost function. The distance is optimised while the coverage is maximised.:

Minimize: $$cost(f_2) = \omega_4(E_{ll}) + \omega_5(1 - Coverage),$$

Ell is described in full in (3). Coverage depicts the network coverage supplied by nodes, whereas 4 and 5 are predetermined constant weights. [2]
The truffle development suite, Web3.js API, and Oraclize API were utilised as development tools to create the Ethereum blockchain in the suggested strategy. Furthermore, smart contracts developed in the Solidity programming language were used to make transactions across all blockchain participants in order to identify decentralised attacks.

Traffic flow analyzer: The traffic flow analyzer dynamically monitors traffic from various IoT devices and collects traffic patterns, including packet and flow-level traffic features such as device utilisation at various time intervals, bandwidth utilisation, total number of sent requests from a device, source of request, and so on..

Model fusion strategy: For the suggested technique to advance, an effective strategy for fusing the separate attack detection models of the different processing agents into a high-performing, efficient attack detection model is required. Each processing agent creates its attack detection model by utilising a deep learning approach to train its private data. To fuse the multiple models, both the manager and the proofing agents need an effective fusion approach. In the area of machine learning, research on multi-model fusion is actively underway..

Given an unlabeled dataset $a = \{a_1, a_2, \dots, a_n\}$ for deep learning model A_k Where n input neuron for the first layer of model $\hat{A_k}$ describes the encoding process as follows:

$h_1 = F(w_1 a + b_1)$

where $\mathcal{F}$ refer to an activation function and w_1,b_1 represent the input layer's weight matrix and the hidden layer's bias vector, respectively. As an activation function, we utilise the sigmoid function, which may be defined as follows::

$F(z) = 1/[1+\exp(-z)$

The first hidden layer output h_1 is utilised as an input to the next hidden layer h_2 for training the network parameters w and b in a deep learning model. The h stands for the second hidden layer feature that was retrieved. To train network parameter W_N, the training method is repeated until the specified Nth hidden layer h1 is reached. The feature extracted at the nth layer of the model Ak is given by hy. We utilise it for convenience of presentation. w = [W_1,W_2, ..., WN], and b = [b_1,b_2, ...,b] to represent the network parameters (weight matrix and bias vector) on the Nth hidden layer of deep learning model A_k

Table 1 lists four kinds of characteristics, their names, and data types based on the NSL-KDD dataset and our recent study. 1-10 and 11-19 indicate the host and time-based traffic aspects, respectively. In addition, the information and certain fundamental characteristics are represented by the numbers 20-31 and 32-41. The rule-setting task covers three primary situations based on the input features:

(1) If the traffic flow is classified as normal traffic by the attack detection model, the SDN controller sets the rules for notifying the switch to allow the traffic flow to continue uninterrupted; (2) If the traffic flow is classified as suspicious or attack traffic, the SDN controller applies a set of rules. Initially, the switch is told to completely block suspicious communications with immediate effect. The source of the assault is blacklisted in the second step. Furthermore, the blacklisted source is updated with the SDN controller at the cloud layer, allowing the blacklisting to be applied at a higher level and preventing the attacker from harming other devices in the IoT ecosystem. The revised rules in the cloud-based SDN controller also aid in preventing scenarios when particular kinds of devices are targeted.
(3) If the attack detection model does not identify the pattern of the traffic flow, the SDN controller advises the switch to restrict the rate of traffic to lessen the consequences of suspicious traffic..[3]

Table 1. Input features to attack mitigation in Io**T**

Category	Features	Data type
Host-based traffic features □	(1) dst_host_count, (2) dst_host_srv_rerror_rate, (3) dst_host_srv_count, (4) dst_host_rerror_rate, (5) dst_host_same_srv_rate, (6) dst_host_srv_serror_rate, (7) dst_host_diff_srv_rat, (8) dst_host_serror_rate, (9) dst_host_same_src_port_rate, (10) dst_host_srv_diff_host_rate	Numeric
Time-based traffic features	(11) srv_diff_host_rate, (12) srv_count, (13) count, (14) diff_srv_rate, (15) serror_rate, (16) same_srv_rate, (17) srv_error_rate, (18) srv_rerror_rate, (19) rerror_rate	Numeric
Content features	(20) logged_in, (21) is_guest_login, (22) is_host_login	Binary
	(23) num_failed_logins, (24) num_outbound_cmds, (25) num_compromised, (26) num_access_files, (27) root_shell, (28) num_shells, (29) su_attempted, (30) num_file_creations, (31) num_root	Numeric
Basic features	(32) Protocol_type, (33) Service, (34) Flag	Nominal
	(35) src_bytes, (36) Duration, (37) dst_bytes, (38) hot, (39) wrong_fragment, (40) urgent.	Numeric
	(41) Land	Binary

ANALYSIS

Results of Clustering A network environment for IoT devices was generated to test the performance of the suggested clustering techniques. It consisted of 100 nodes that were produced at random and spread in a 2-D network. MATLAB 2018a was used since it provided a trustworthy environment for clustering methods, as well as a simple simulation of algorithms that allowed the results to be compared. The GAs parameters used in this scenario are listed in Table 1..

Table 1. GAs Parameter Settings**.**

GAs Parameters	**Value**
Population Size	30
Selection Type	Proportional Selection
Recombination Percentage	0.1
Crossover Percentage	0.5
Crossover Type	One-Point
Mutation Percentage	0.4
Mutation Rate	0.05
Generation Size	500

Through APIs, the client application may access the state database and conduct different queries, such as put, get, and delete. The deployment of a REST server to directly offer RESTful APIs that may be used while using a web client or a

virtual device was used to define various blockchain functionalities. The user may use GET or POST to submit different transactions through HTTP queries to appropriate APIs. The Fabric client application is hosted on the REST server and communicates with the HLF network via the Google Remote Procedures Calls (gRPC) technology. A copy duplicate of the ledger is held by each peer in the network.

The ledger is divided into two sections: a transaction log and a list of all recorded state changes. The versioned key-value pairs are also included in the state data. All changes to the state database are stored in the ledger in chronological order, and the blocks are cryptographically linked. The orderer node uses the PBFT algorithm to verify ledger consistency. The Execute-Order-Validate and Commit transaction models are supported by the HLF framework..[2]

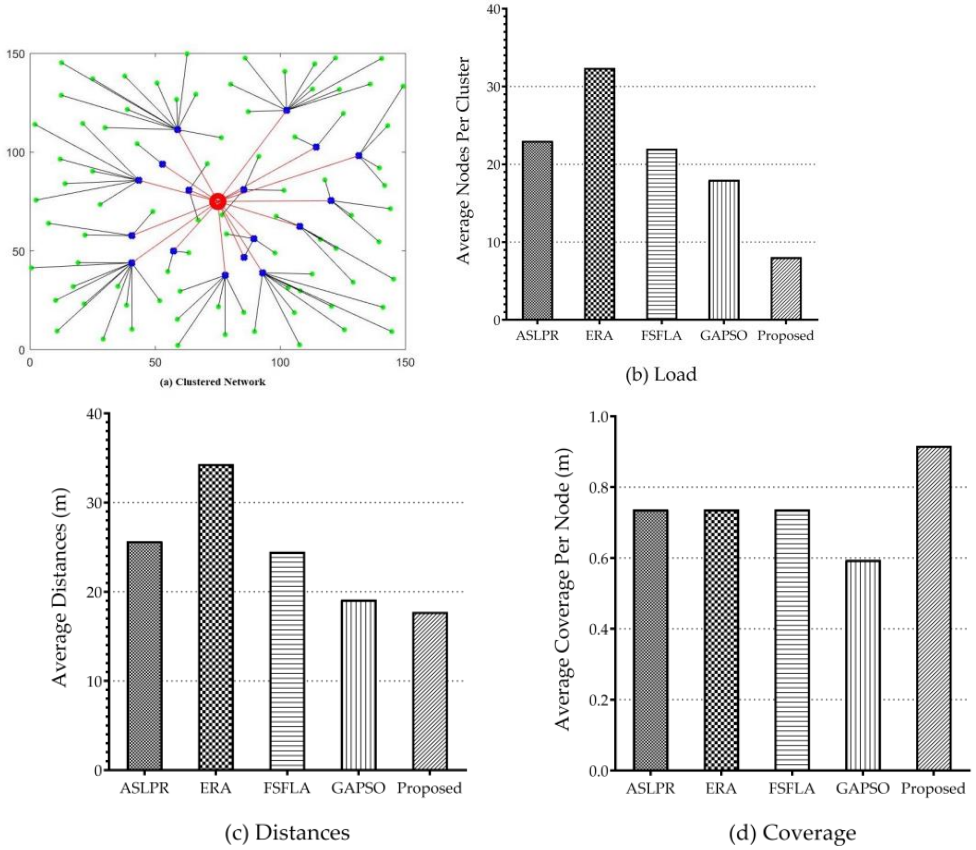

Figure 2. Performance of the proposed clustering algorithm. (a): clustered network and CH positions. (b–d): benchmarked performance in terms of load, distances, and coverage, respectively

The findings of the performance assessment of the proposed BlockSecIoTNet architecture are presented in this section. Mininet [30] was utilised as an emulation environment to simulate the open vSwitches and different functional nodes in the proposed architecture, such as IoT devices. We set up Mininet on the Linux server with 15 PCs, each with 64GB DDR3 RAM and an Intel i7 CPU. To integrate machine learning classification for evaluating traffic behaviour for attack detection, we employed POX as a controller. For the fog and edge nodes, the controllers were executed in separate VMs housed on a Linux server. As a cloud server, the Amazon EC2 cloud data centre was employed. We employed the Ethereum blockchain technology to allow decentralisation in our design, and an oracle was deployed in the private chain by deploying the Ethereum Bridge in the broadcast mode.

The manager gave cash incentives to the processing agents and accessed the attack detection model shared by the awarded agent based on the assessment findings. We delivered private data from the NSL-KDD dataset, which is an intrusion detection dataset, to each processing agent in the simulations. The private data included both regular and attack traffic patterns from IoT devices. We also assumed that 10 fog nodes served as processing agents, while the other five served as proofreading agents. Each processing agent has 500 distinct types of training data and 500 different types of testing data. The management (cloud server) sent the identical dataset to each proofing agent for verification. Table 2 shows the distribution of data among the processing and proofreading agents..

Table 2. Distribution of data among agents

Agent	No. of training instances	No. of testing instances
Processing agent	500	500
Proofing agent	2000	500

Accuracy, Positive Predictive Value (PPV), Detection Rate (DR), F-score, Mathew Correlation Coefficient (MCC), Detection Time (DT), and the area under the Receiver Operating Characteristic (AUC) curve, which are the most important evaluation metrics for attack detection, were used to evaluate the performance of the proposed architecture. We initially analysed and compared the proposed decentralised architecture to two existing architectures, namely the cloud-based centralised model and the fog-based distributed model, using these measures. Second, we looked at the performance of all three designs in the context of various attack scenarios.

.

Comparison of performance of different architectures: The suggested decentralised architecture was compared to two distinct architectures: a cloud-based centralised model and a fog-based distributed model. At the Amazon EC2 cloud data centre, we used a deep learning methodology for threat detection in the centralised architecture. Following that, we used the deep learning methodology to execute attack detection in the distributed architecture at the fog node. In terms of standards evaluation metrics, Fig. 3 shows a comparison of the proposed decentralised architecture's performance with the centralised and distributed architectures, clearly demonstrating that the distributed architecture outperformed the centralised architecture and that the decentralised architecture outperformed both the centralised and distributed architectures. Attack detection is performed in the centralised architecture by processing data from IoT devices at the cloud server, whereas attack detection is performed at the fog level by processing data from IoT devices connected to each fog node, which shares the detection load among several fog nodes and improves attack detection performance.

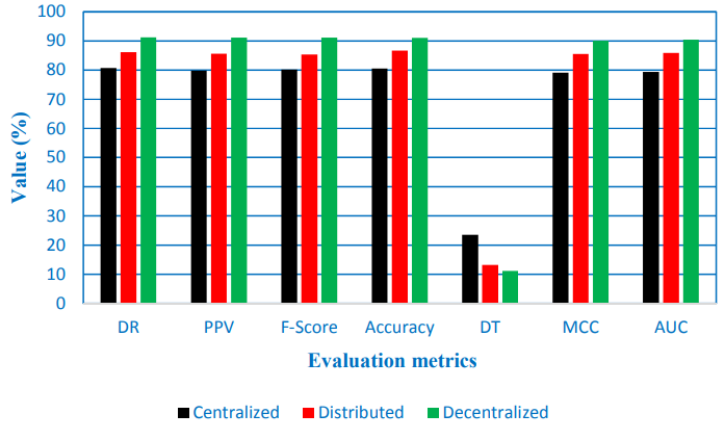

Fig.3 . Comparison of the performance of the different architectures

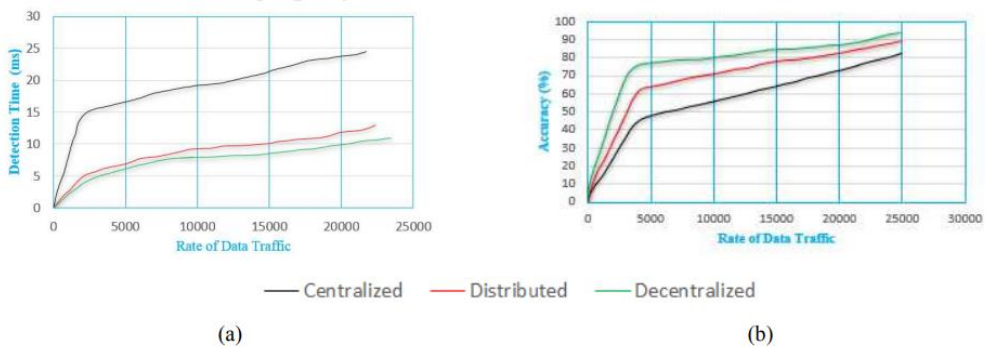

Fig. 4. Attack detection performance of different architectures with a varying rate of data traffic versus: a) detection time: b) accuracy

In our decentralised design, however, the attack detection model at each fog node is constantly updated via blockchain technology, which increases attack detection speed. In addition, Fig. 4(a) depicts the relationship between the detection time and the total amount of data flow. The detection time in all three models constantly rises as data traffic grows. The decentralised architecture, on the other hand, consistently outperformed the distributed and centralised structures in terms of detection time. This might be because, in a decentralised architecture, the fog node develops the attack detection model and modifies the flow rule in the SDN switch at the edge for attack detection, which puts it closer to the IoT devices and reduces detection time. Similarly, Fig. 4(b) depicts differences in accuracy as a function of overall data flow. In all three designs, the increased amount of data flow gives more data for the deep leaning classification process in attack detection, resulting in more accurate attack detection. However, unlike distributed and centralised designs, the decentralised architecture leverages blockchain technology to constantly update the attack detection model at each fog node, resulting in more accurate attack detection. Overall, the decentralised architecture outperforms both centralised and distributed

architectures in detecting security attacks in the IoT network, implying that decentralised attack detection using blockchain technology is an effective approach for detecting attacks in smart IoT ecosystems such as self-driving cars, where more precise and real-time decisions are required..[3]

CONCLUSION

Based on distributed blockchain technology, this article presents a multi-layer security architecture for IoT devices operating on multi-hop cellular networks. The created approach offers a viable method for establishing a decentralised blockchain application for the security of a cellular-enabled IoT network. The hybrid self-clustering EC technique is designed to split the IoT network into clusters in order to create multi-layer structure and increase network lifespan. The way the blockchain-based model may assist to enhance IoT system authentication and authorisation is explained, as well as the details of the system implementation. For implementation and verification, the model recommends the open-source HLF blockchain. Three new contributions to IoT security have been made by the proposed architecture. First, the suggested architecture employs SDN to constantly monitor and analyse traffic data throughout the whole IoT ecosystem, addressing the problem of data unavailability in vulnerability detection and ensuring the best possible security protection. Second, the design makes use of Blockchain technology, which allows for decentralised attack detection and so avoids the single point of failure issue that centralised and distributed systems have.

REFERENCES

1. CH. V. N. U. BHARATHI MURTHY (2017) “Blockchain Based Cloud Computing: Architecture and Research Challenges” Received October 23, 2020, accepted November 4, 2020, date of publication November 9, 2020, date of current version November 23, 2020. Digital Object Identifier 10.1109/ACCESS.2020.303681
2. Sultan Algarni (2016) “Blockchain-Based Secured Access Control in an IoT System” Blockchain-Based Secured Access Control in an IoT System. Appl. Sci. 2021, 11, 1772. https://doi.org/ 10.3390/app11041772
3. Shailendra Rathore (2015) “BlockSecIoTNet: Blockchain-Based Decentralized Security Architecture for IoT Network” PII: S1084-8045(19)30224-3 DOI: https://doi.org/10.1016/j.jnca.2019.06.019
4. Muhammad Adnan Khan, (2017) “A Machine Learning Approach for Blockchain-Based Smart Home Networks Security” June 2021 IEEE Network 35(03):223-229 DOI:10.1109/MNET.011.2000514
5. Amit Kumar Singh (2013) SmartNoshWaste: Using Blockchain, Machine Learning, Cloud Computing and QR Code to Reduce Food Waste in Decentralized Web 3.0 Enabled Smart Cities February 2022 Smart Cities 5(1):162-176 DOI:10.3390/smartcities5010011
6. Ashok Gupta (2016) “Cloud Computing Security using Blockchain” © 2019 JETIR June 2019, Volume 6, Issue 6 www.jetir.org (ISSN-2349-5162)
7. Wenjuan Li (2018)“Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions” Li et al. Journal of Cloud Computing: Advances, Systems and Applications (2021) 10:35 https://doi.org/10.1186/s13677-021-00247-5
8. Yiannas, F. A new era of food transparency powered by blockchain. Innov. Technol. Gov. Glob. 2018,12, 46–56. [CrossRef]
9. Marin, M.P.; Marin, I.; Vidu, L. Learning about the reduction of food waste using blockchain technology. arXiv2021arXiv:2101.02026.
10. Dey, S.; Mondal, K.; Nath, J.; Nath, A. Advanced Steganography Algorithm Using Randomized Intermediate QR Host Embedded With Any Encrypted Secret Message: ASA_QR Algorithm. Int. J. Mod. Educ. Comput. Sci. 2012,4, 59–67. [CrossRef]
11. Huang, P.C.; Chang, C.C.; Li, Y.H.; Liu, Y. Efficient QR code secret embedding mechanism based on Hamming code. IEEE Access 2020,8, 86706–86714. [CrossRef]
12. De Donno, M.; Tange, K.; Dragoni, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. IEEE Access 2019,7, 150936–150948. [CrossRef]
13. Qi, Q.; Tao, F. A smart manufacturing service system based on edge computing, fog computing, and cloud computing. IEEE Access 2019,7, 86769–86777. [CrossRef]