EasyChain: An IoT-Friendly Blockchain for Robust and Energy-Efficient Authentication

Anand K. Bapatla [1], **Deepak Puthal** [2], **Saraju P. Mohanty** [1,*], **Venkata P. Yanambaka** [3], **and Elias Kougianos** [4]

[1] *Dept. of Computer Science and Engineering, University of North Texas, Denton, TX, USA*
[2] *Dept. of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE*
[3] *Dept. of Computer Science, Texas Woman's University, Denton, TX, USA*
[4] *Dept. of Electrical Engineering, University of North Texas, Denton, TX, USA*

Correspondence*:
Saraju P. Mohanty
saraju.mohanty@unt.edu

ABSTRACT

The Internet of Everything (IoE) is a bigger picture that tries to fit the Internet of Things (IoT) that is widely deployed in smart applications. IoE brings people, data, processes, and things to form a network that is more connected and increases overall system intelligence. Further investigating the IoE can really mean creating a distributed network focusing on edge computing instead of relying on the cloud. Blockchain is one of the recent distributed network technologies which by structure and operations provide data integrity and security in trust-less P2P networks such as IoE. Blockchain can also remove the need for central entities which is the main hurdle for wide adoption of IoT in large networks. IoT "things" are resource constrained both in power and computation to adopt the conventional blockchain consensus algorithms that are power and compute-hungry. To solve that problem, this paper proposes EasyChain, a blockchain that is robust along with running on a lightweight authentication-based consensus protocol that is Proof-of-Authentication (PoAh). This blockchain based on lightweight consensus protocol replaces the power-hungry transaction and block validation steps and provides ease of usage in resource-constrained environments such as IoE. The proposed blockchain is designed using the language Python for easy understanding of the functions and increased ease of integration into IoE applications. The designed blockchain system is also deployed on a single-board computer to analyze its feasibility and scalability. The latency observed in the simulated and experimental evaluations is 148.89 ms which is very fast compared to the existing algorithms.

Keywords: Internet-of-Everything (IoE), IoE Security, Internet of Things (IoT), IoT-Device Security, Blockchain, Distributed Ledger, Consensus Algorithm, Energy-Efficient Cybersecurity

1 INTRODUCTION

Many definitions were given for the Internet of Things (IoT) since the term was coined in 1999 (S. P. Mohanty et al., 2020). A typical IoT architecture consists of devices that are coined as "things" that are connected over a network using different Information and Communication Technologies (ICT's) and perform resource-intensive operations in the cloud. Unique identification is one of the main characteristics of the things along with the capability to connect to the Internet. Unique identification can either be a MACID assigned to Network Interface Card (NIC) or the IP address assigned by the network to each individual device that is connected. IoT architecture is being used in many applications that can range from smart healthcare to industrial IoT and smart cities (Castanho et al., 2019; Corbett et al., 2018; Shahzad and Kim, 2019; Xu et al., 2022; Mitra et al., 2022). Combining these IoT networks with people and processes create Internet of Everything (IoE). In Healthcare Cyber-Physical System (H-CPS), which is a very complex environment, many IoT networks are used at supply chains, medical centers, care centers, etc. to continuously monitor and provide better care to patients. According to the Health Insurance Portability and Accountability Act (HIPPA), such sensitive healthcare information should be handled with high data privacy and security. As the number of connected devices is increasing day-to-day and implementing robust security mechanisms at the expense of higher computation and power requirements is not a feasible solution for IoE environments. The lack of such robust security systems in place has opened doors for attackers to remotely gain unauthorized access to the systems (Mohanty et al., 2020; Alfandi et al., 2020).

IoT architecture is independent of the communication protocol stack such as TCP/IP and is powered by many lightweight protocols which can accommodate the low bandwidth IoT requirements (Dorri et al., 2017). Things in IoT are responsible for collecting the sensory data and transmitting it to the end devices which typically are single-board computers (SBC) with little higher computational and storage capabilities compared to things. Collection and communication of these environmental data is one of the major concerns in IoT architecture and is facilitated by different middleware technologies (Moreno et al., 2017). The edge layer which consists of Edge Data Centers (EDCs) will act as real-time data processing units and provide emergency data processing (Zanella et al., 2014; Puthal et al., 2016; Zhaofeng et al., 2020) capabilities for IoT deployments due to its decentralized nature. Several applications which include military, industrial IoT, etc. can be implemented using EDC's integrated IoT architecture with low power consuming and resource-constrained devices. Data collection and secure transfer are the important aspects of such architectures implemented in critical applications. Many security algorithms exist which involve both symmetric and asymmetric cryptography techniques, due to the lower power and computational resources at IoT systems symmetric encryption is widely adopted that performs 1000 times better than asymmetric cryptography (Puthal et al., 2017). Symmetric key encryption is less secure and cannot be used for non-repudiation of the devices connected to the network. As the same shared key is used for both encryption and decryption, it can be argued the receiver itself encrypted the message making device authentication not possible. In a typical IoE architecture, the cloud layer is an integral part that is capable of processing large amounts of data with larger computational power (Mukhopadhyay et al., 2021). Cloud services utilized in architecture form a centralized system and can introduce issues like latency and Single Point of Failure (SPOF).

The cryptography operations performed during the communication, and the central entity can be replaced by leveraging a blockchain which can resolve the requirements of a central authority for reaching a consensus among the distributed participants. The main component of blockchain is a decentralized ledger which is used to store the data and timestamped transactions chronologically between the un-trusted distributed entities. Every entity participating in a P2P network has a copy of the entire or part of the ledger. A special type of node called miners present in the P2P network are responsible for validating transactions and performing consensus before storing them in the immutable ledger. A blockchain is cryptographically anchored and tamper-proof and is a record of the different transactions that occurred among the participants in the network (Puthal et al., 2018).

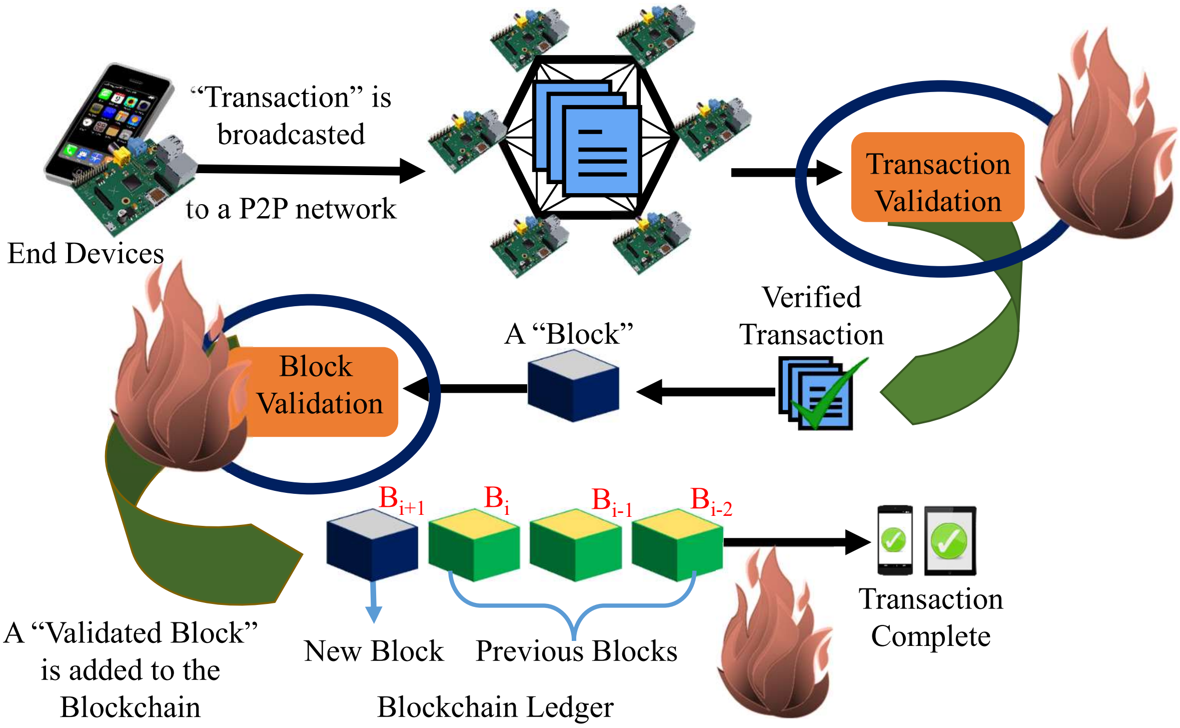


Figure 1. Block generation, validation and addition in general Blockchain. Flames represent the resource intensive tasks during the process.

A central entity is not present in a blockchain architecture and a consensus algorithm is used to secure the distributed ledger and maintain consensus among the nodes in the network (Zyskind et al., 2015; Qu et al., 2021). A cryptographic hash is used to connect the previous blocks to the new blocks in the distributed ledger. This helps secure the ledger from anyone tampering with the transactions. Some widely used blockchain consensus algorithms are Proof-of-Work (PoW), Proof-of-Stake (PoS), and Proof-of-Activity (PoA). But the existing algorithms require high computational capabilities and resources which are not available in IoT architectures.

The process of block generation is shown in Fig. 1. The figure shows the process which requires high power and resources during the block validation and addition. The devices form blocks with multiple transactions and broadcast to the network. The miners in the network will validate the transactions, which consumes more power and requires high resources. Once the transactions are validated, the reverse hash of the block is calculated, which requires high resources. Local storage of the devices is also a bottleneck in the case of IoT architectures. All these issues are addressed in the proposed Proof-of-Authentication (PoAh), a lightweight blockchain consensus algorithm for IoT architectures. The current paper also implements a blockchain called EasyChain which operates on a PoAh consensus mechanism and can be seamlessly integrated into IoT.

The rest of the paper is organized as follows: Section 2 discusses contributions and novel solutions proposed from current work. Section 3 discusses the blockchain as a security primitive for the IoE. Section 4 surveys existing consensus mechanisms and their adaptability to IoE cyber-physical systems. Section 5 discusses insights into the proposed EasyChain and its software architecture. Section 6 describes the access control mechanism implemented for the proposed EasyChain. Section 7 discusses the novel consensus protocol Proof-of-Authentication (PoAh) proposed in EasyChain. Section 8 presents experimental evaluation and validation of the proposed EasyChain. Section 9 includes a discussion on different claims of the proposed PoAh consensus algorithm followed by Section 10 concludes the paper and presents possible future research.

2 CONTRIBUTIONS OF THIS WORK

2.1 Problem Definition

An estimated 18 billion devices will be connected to the network across the globe. The majority of the connected devices will be sensors and other smart devices constantly monitoring the environment (Novo, 2018; Huang et al., 2017). Blockchain is one of the most promising solutions to be integrated into IoT for decentralized security. It is predicted that a blockchain can:

- Maintain the device authentication and immutability (Nayak and Dutta, 2017).
- Maintain the integrity of data collected by IoT devices making it difficult to tamper with the data added to the ledger (Kshetri, 2017).
- Decentralize the nodes making them more reliable and scalable (Nayak and Dutta, 2017).
- Leverage edge computing paradigm to provide near real-time data operations in the distributed network.
- Ensure an adversary-free network and eliminates false data injection in the network by providing device authentication which can be a major contribution to healthcare systems.
- Robust and fine-grained access control mechanism for accessing processed data from networks.

The requirements of an IoT architecture differ from those of cryptocurrencies making the integration difficult. Multiple issues must be addressed in the blockchain IoT integration (Wang and Malluhi, 2019):

- Time taken to validate and add the block to the ledger (Xin et al., 2017).
- Improved infrastructure to support high bandwidth for IoT devices (Kuzmin, 2017).
- Ensuring power-constrained consensus models to be deployed into IoT architectures.
- Easy integration to existing IoT architectures that can help in wide adoption.
- Easy-to-use functions reducing the computation requirements at the device level to generate blockchain transactions.

Researchers in academic and industry areas are focusing more on integrating blockchain to IoT architectures due to the promise of solving security and data integrity issues of IoT (Ouaddah et al., 2016; Novo, 2018). This paper presents one such blockchain solution for IoE architecture with a novel lightweight consensus algorithm. The proposed solution can be easily integrated into resource-constrained IoT systems along with providing both data and device security.

2.2 Proposed Novel Solution

Integrating a blockchain consensus algorithm into an IoT architecture is a highly challenging task due to the resource-constrained devices in the IoT network. But IoT devices are deployed in environments where they are not constantly monitored. So, IoT can benefit from a decentralized network and consensus algorithm provided by the blockchain. As a solution, a lightweight consensus algorithm, PoAh is presented in the paper and a blockchain along with EasyChain which integrates the PoAh into an IoE architecture is also proposed. Novel aspects of the proposed blockchain are as follows:

- PoAh adds a cryptographic authentication mechanism for both data and devices in the P2P network.
- EasyChain is proposed for resource-constrained real-time IoE architectures.
- Proposed blockchain mechanism EasyChain can serve requirements for private blockchain solutions.
- A robust access control mechanism adaptable in private use cases is proposed.
- Proposed EasyChain is evaluated as a solution for different use cases related to IoT healthcare systems.
- Finally, EasyChain is validated with both simulated and experimental setups using a real-time test bed for performance evaluation.

3 BLOCKCHAIN AS A SECURITY PRIMITIVE FOR IOE

There have been many applications and architectures of IoT since the term was first coined. IoT architectures were widely adopted across diverse areas including Smart Cities, Industries, Home automation, and Smart Healthcare (Misra et al., 2021). Among these applications, the IoT has the most potential in solving many issues in the Healthcare industry. Many smart healthcare IoT architectures were designed and deployed across the world. This also helps in monitoring patient health remotely and administering the drugs if necessary. IoT in the smart healthcare industry handles data related to patients. Things constantly monitor the patients and transmit the data to the cloud (Kumar et al., 2020; Shahzad and Kim, 2019). Privacy and security of such data must be given the highest priority. There are many threats possible in an IoT environment, and in specific, healthcare. A simple threat can potentially endanger the life of a patient. Many smart applications can be potentially targeted by the attackers to gain access to a household, through a patient tracking device or access patient data (Hassija et al., 2019).

An access attack or an advanced persistent threat (APT) can grant the attacker access to the IoT network (Hassija et al., 2019). Detecting the attacker in the network is challenging once the attack is successful and grants access, information in the network can be stolen by the attacker. The wearable or implantable devices present in the network constantly transmit the data to the cloud storage which can be monitored by the attackers. A data transit attack is another vulnerability through which an attacker can gain access to the data being transmitted to the storage. Another potential issue with IoT applications is the power supply. Implantable Medical Devices (IMD) are required to work for long periods of time before requiring a battery change. IoT architectures used in such applications must be designed with low power-consuming devices and protocols. There are also many attacks that can potentially drain the battery of an IoT device by running an injected code in a loop (Hassija et al., 2019). Blockchain can be a potential solution for such issues mentioned above. Table 1 presents such challenges present in the IoT architectures and how blockchain can act as a potential solution for such challenges.

4 RELATED PRIOR WORKS

As one size doesn't fit all, different consensus protocols are proposed for various applications. The most commonly used consensus protocol is Proof-of-Work (PoW) which works based on the hashcash CPU cost-function proposed in (Back, 2002). In this consensus protocol, different nodes in the network race to solve a cryptography hash function to find the right nonce. Node finding the right nonce will be given the opportunity to add a new block for which incentives will be awarded. PoW is widely used in cryptocurrencies, however high computational requirements make it not suitable for resource-constrained IoT environments.

Table 1. Blockchain as Potential Solution for IoT Challenges.

Category	Challenges in IoT Architectures	Blockchain as a Potential Solution
Privacy and Security in IoT architecture	Data stored on IoT devices is vulnerable to attacks.	A secure blockchain can store the data anonymously and maintain privacy.
	Data can be spoofed in IoT devices.	Device authentication consensus algorithms can be used to secure the IoT environments.
Computational	"Things" in an IoT environment are not computationally intensive.	All the computations in blockchain are offloaded to miners or trusted nodes.
Power	IoT devices are deployed in remote locations possibly operating on a battery.	Blockchain increases the security and privacy of the environment and offloads computations to the trusted nodes and miners which reduces power load on battery.
Form factor	IoT devices in some cases are required to be smaller.	In blockchain enabled IoT architectures, the "things" only has the sensors and communication module reducing the overall form factor of the devices.

Proof-of-Stake (PoS) (King and Nadal, 2012) is another popular consensus mechanism next to PoW which mainly uses the amount of stake and coin age as the parameters to choose the miner instead of the computational capacity like PoW. This removes the need for high computational requirements and increases the throughput of the network. However, like PoW, PoS is more popular in cryptocurrency networks, but the concept of stake is not relevant for IoT systems. Delegated Proof-of-Stake (DPoS) which is a variant of PoS is proposed in the BitShares project. In DPOS a certain number of witnesses or block producers are selected by the user votes. Users pool their tokens in a staking pool and elect a delegate to participate in the block production on their behalf. The transaction reward received will be distributed among the winning delegate and users who elected the delegate. It is based on the reputation of the node and like PoS works on the principle of stakes. Even though DPOS is faster than PoW and PoS, it is more centralized, and dependency on monetary aspects makes it not a good candidate for IoT systems. Proof-of-Importance (PoI) is a variant of PoS in which the winning node is selected based on reputation computed using multiple factors like the number of transactions validated correctly along with the staking coins. NEM blockchain (NemProject, 2018) uses the PoI consensus mechanism. Like PoS and DPoS, PoI also depends on stakes.

Proof-of -Elapsed Time (PoET) is another consensus protocol proposed by Intel in Hyperledger Fabric (Olson et al., 2018). It performs the same operations as PoW but the winning node is chosen based on the expiration of time allocated to that node instead of resource-intensive problems. Random wait time values will be assigned to each node through Trusted Execution Environment (TEE). Even though it provides higher throughput but as it is specially designed for private networks, the mechanism is making the network centralized and heavily depends on Intel tools like Software Guard Extensions (SGX).

Proof-of-Activity (PoA) is a combination of PoW and PoS. In the first step, miners will perform a complex cryptography puzzle to create a blank template block with only header information and mining reward address, and doesn't have any transactions. In the later step, PoS mechanism is applied to find the validators to check the block and add it to the network. Once a valid block is added, transactions will be recorded onto the newly created block. PoA has high energy consumption and latency which makes it not a viable solution for IoT systems.

Practical Byzantine Fault Tolerance (PBFT) consensus protocol proposed in (Castro, 1999) to solve Byzantine General Problem (Lamport et al., 1982) in a distributed system. In this consensus protocol, all the nodes are ordered to form a primary or leader node and secondary or backup nodes and participate in the consensus mechanism. The goal of PBFT is to reach a consensus in the network even with a certain threshold of malicious nodes participating in the network. This threshold must be not greater or equal to one-third of the nodes in the network. Although it is robust, it doesn't provide scaling for large networks like IoT systems and results in large network overhead. Different variations of PBFT are also proposed Multi-Layer PBFT (Li et al., 2021) which significantly reduces the network overhead with the increase in layers but sacrificing the latency requirement. Federated Byzantine Fault Tolerance (FPBFT) which is used in Stellar Consensus Protocol proposed in (Mazieres, 2015). Ripple Consensus protocol in (Chase and MacBrough, 2018) works on low latency Byzantine Fault Tolerance mechanism to improve the latency and reach consensus even before a full agreement of the network.

Proof-of-Vote (PoV) proposed in (Li et al., 2017) is designed for consortium blockchain and works based on the decentralized arbitration of votes. Proof-of-Trust is another protocol proposed in (Zou et al., 2018) which selects the validators based on the trust values of the participants and makes use of RAFT leader election and Shamir's secret sharing algorithms to reach consensus.

A credit-based PoW mechanism is proposed in (Huang et al., 2019) which performs PoW and the difficulty of the puzzle changes dynamically based on the honesty of the node. With the honest node, the PoW puzzle takes less time compared to the node with dishonesty. Another reputation-based consensus Proof-of-Reputation-X (PoRX) (Wang et al., 2020) considers the nodes as per the positive contributions provided thereby reducing the need for ASIC mining and consuming less power.

Table 2. Comparative perspective of PoAh with other related works.

Consensus Algorithm	Blockchain Type	Mining	Prone To Attacks
Proof-of-Work(PoW)(Back, 2002)	Permission-Less	Based on Computational Power	Bribe attack, Sybil attack, 51% attack
Proof-of-Stake(PoS)(King and Nadal, 2012)	Permission-Less	Validation	DoS, Sybil attack, Nothing at stake
Ripple (Chase and MacBrough, 2018)	Permissioned	Vote Based Mining	DoS attack, Sybil attacks
Proof-of-Vote(PoV)(Li et al., 2017)	Consortium	Vote Based Mining	-
Proof-of-Trust(PoT)(Zou et al., 2018)	Permissioned	Probability and Vote Based Mining	DDoS attack
Proof-of-Reputation-X (PoRX) (Wang et al., 2020)	Permission-Less	Reputation Based	-
Proof-ofAuthentication(PoAh)(Current Paper)	Permissioned	Authentication	Currently Testing

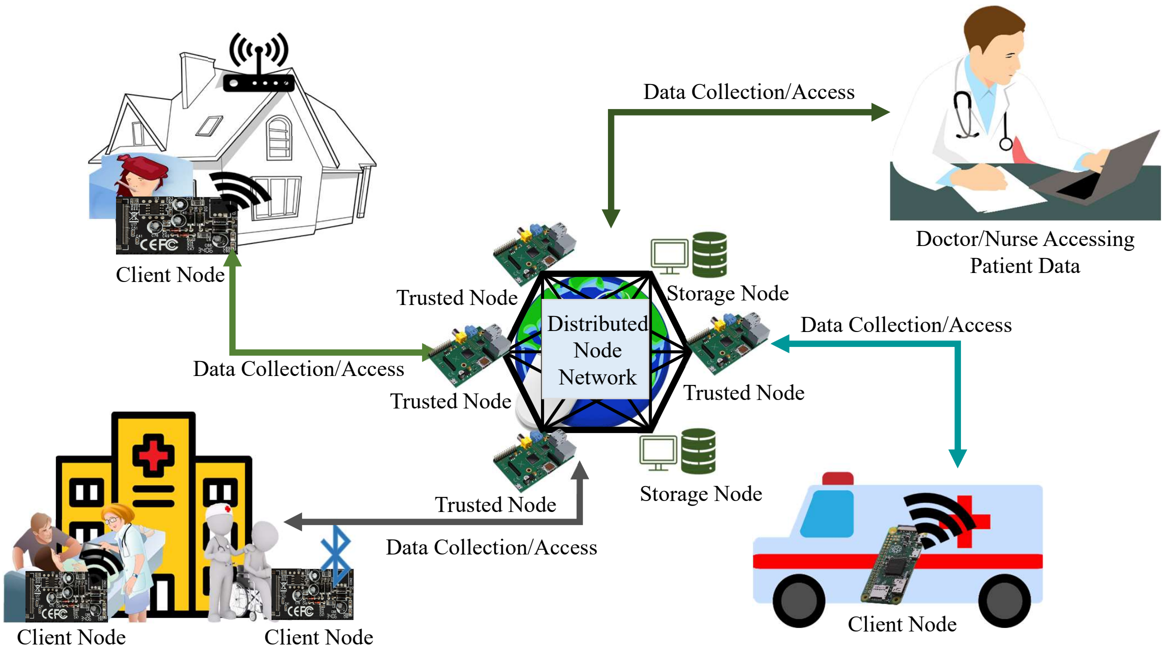


Figure 2. Proposed Blockchain can have applications in various IoT driven systems.

5 THE PROPOSED NOVEL BLOCKCHAIN - EASYCHAIN

The proposed blockchain architecture uses the novel PoAh consensus algorithm. PoAh consensus algorithm authenticates the devices that are transmitting the data to the network and adds the respective data to the blockchain. PoAh is also significantly better compared to the existing consensus algorithms in various aspects such as latency, scalability, and power consumption. There are three major entities in the PoAh: the trusted node, the client node, and the storage node. The trusted node, as the name suggests, is a node from the network of trusted devices that is responsible for authenticating the other devices. The client nodes are in the field, or at the user end collecting the information. Storage nodes are devices with large storage capabilities which will store the entire trail of transactions in the network as the client and trusted nodes have limited storage capabilities. Fig. 2 shows a scenario of the proposed blockchain by taking the Internet of Medical Things (IoMT) as a use case. As shown in the figure, the patient data is being collected by IoMT devices. The patient's location or state does not constrain the blockchain. The patient can be at their home, a care facility, a hospital or transported in a vehicle. In all these scenarios, the patient is constantly monitored by the IoT devices, and the data is transmitted to the P2P network using PoAh consensus algorithm

The IoMT devices that are with the patient are the client nodes. The requirements of an IoMT device to act like a client device in PoAh are basic cryptographic functionality and communication capabilities. These two functionalities are available in most off-the-shelf components currently. The client node monitors the patient's vitals and constantly transmits the data to the trusted nodes network. Resource-constrained client nodes do not have the necessary memory to store the complete blockchain. As designed, EasyChain performs data transactions, so there is no need for the client nodes to store the entire trail of previous transactions, in contrast to financial applications where double spending must be verified. With only limited memory, client nodes can store only the most recent transactions.

The devices in the trusted node network are responsible for authenticating both client node device and the transaction data sent by the client node. The trusted node network has access to the identities of the

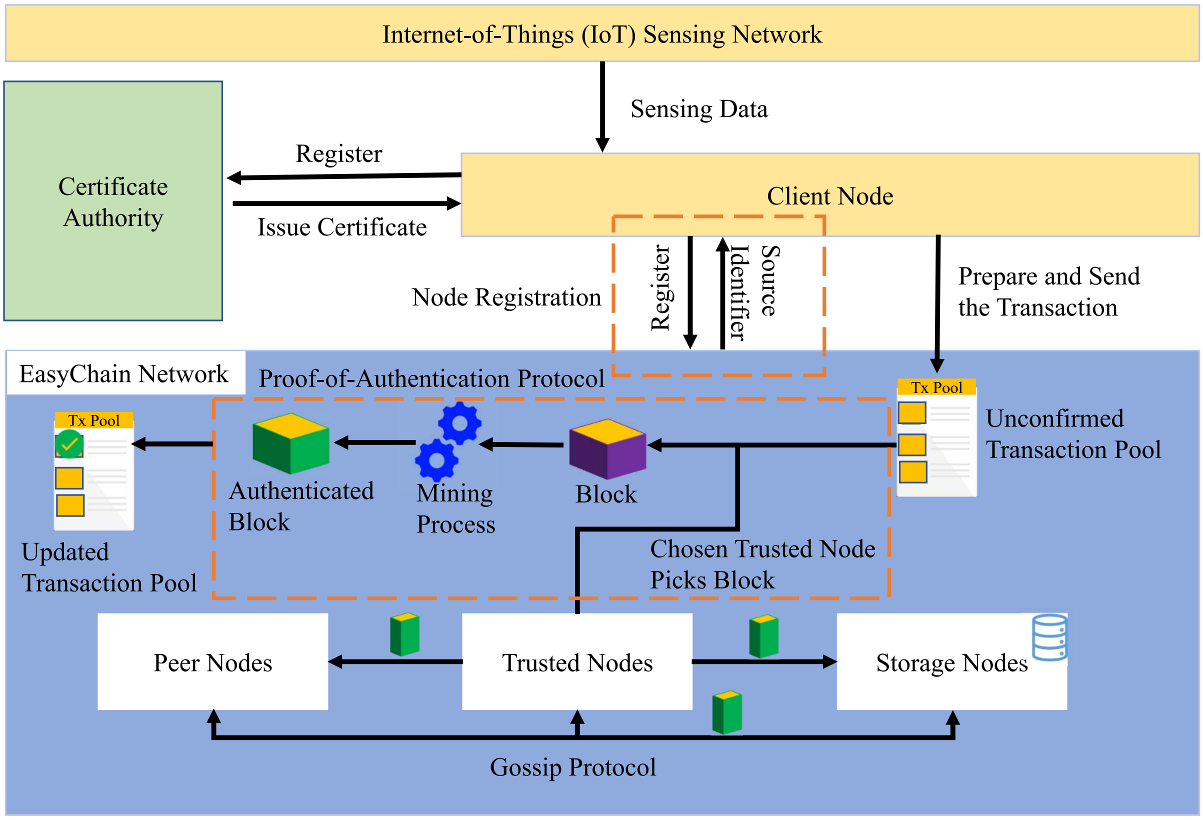


Figure 3. Software Architecture of Proposed EasyChain.

client devices present in the network. An off-the-shelf single-board computer can be used as a trusted node. As the proposed EasyChain is designed for private networks, the participating distributed entities in the system will initialize some of the participating nodes as trusted nodes by assigning a trust value greater than the threshold. Trust values are updated based on each block authentication, as discussed in Section 7

Storage nodes in the network have large storage capabilities compared to trusted or client nodes. As the transactions accumulate, the size of the data increases, and handling such large amounts is not possible with resource-constrained single-board computers. Storage nodes help in retaining the information of entire transactions and data trail which is helpful in accessing information from the network. Multiple storage nodes are deployed and maintained by distributed entities in the network. In a healthcare setup these entities can be a network of hospitals, Emergency Medical Services (EMS) Electronic Health Record (EHR) systems, etc. As the data is available at multiple locations, the proposed architecture is resistant to SPOF and achieves higher system availability.

Once the blocks are added to the blockchain it gets transmitted to the storage nodes. A nurse practitioner or a doctor can access the data from the storage nodes. This makes it easier for the doctor to monitor the patient's vitals remotely with very low latency. The blocks take around 400 milliseconds to get authenticated and added to the blockchain with high traffic. This ensures low-latency transactions and makes it easier for the doctor to access patient data. The patient's vitals can be assessed by the doctor while transported to the hospital in an ambulance and develop a treatment strategy accordingly. The proposed EasyChain mechanism can be divided into three steps: Initial registration of the client nodes, Generation and processing of transactions, and a Robust access control mechanism to ensure secure access to patient data. The software architecture of the proposed EasyChain is shown in Fig. 3.

Algorithm 1 Registration of New Nodes Into EasyChain Network

Input: Each Node will have its identity associated with MACID, Source ID and their own assigned Private ($P\gamma K$) and Public keys ($P\upsilon K$). Port number (Port$_{num}$) at which the client application is running.
Output: Node list at all the network nodes will be updated with newly added node.

```
for Every New incoming node N into network do
    A unique source ID (SID) which is random and unique to this node is generated.
    RSA Public key (PυK_N) and Private Keys (PγK_N) are generated and assigned to this node.
    Private Key generated PγK_N ← rsa.generateNewKey(public exponent, key size)
    Public Key generated PυK_N ← PγK_N.getPublicKey()
    RSA Private PγK_N and public PυK_N keys are persisted in client node secure storage location.
    Public key file ← writePublicKey(PυK_N, fileName)
    Private key file ← writePrivateKey(PγK_N, fileName)
    New node is registered and broadcast to all network nodes
    registerAndBroadcastNode(Port_num,MACID,SID,PυK_N)
    for Each Node N_i in Existing Node List do
        Node list of N_i is updated with new node information
        NodeList_i.append(Node_N(Port_num,MACID,SID,PυK_N))
    end for
    NodeList_N ← getNodeListOfExistingNodes()
    Run Consensus and copy the longest acceptable chain to new node N
    Chain for Node N Chain_N ← getLongestAcceptedChain()
    Return SID
end for
```

During the initial step, every client node should register in EasyChain network. Each node in the network is assigned unique private and public RSA cryptography keys. Assigned public key $P\upsilon K_N$ and private key $P\gamma K_N$ are stored at the secure file location of the client node. The client node will send MACID, and a randomly generated unique id called Source ID (SID). A node list is maintained by each participating node in the network, which helps in peer discovery for new nodes. Once all the existing nodes are updated with the new node information, a copy of this node list from the existing node will also be copied to the newly added node for the discovery of other existing nodes by the new node. Along with that, the chain information is also copied to the new node N during the initialization/registration phase. Detailed steps of new node registration into the network are shown in Algorithm 1

Once the client node is registered into the network, it can generate transactions and share data within the network. The generated transaction from the edge client node will be hashed using the SHA-256 hashing algorithm and is used to generate the digital signature using the private key of the edge client node. The digital signature generated will then be appended to the transaction data along with the MACID. The digital signature is used as the primary authentication step of the Proof-of-Authentication algorithm whereas MACID is for secondary authentication. Once the transaction is created by the edge client node, it will be broadcast to the entire network and will be added to the pool of unconfirmed transactions. Trusted nodes in the network will pick up the transactions which are yet to be confirmed from the unconfirmed transaction pool. The trusted node then computes the hash of transaction data using the same hashing algorithm (SHA-256) and using the public key of transacting node hash which is retrieved from the digital signature. Both these hashes are then compared to check the integrity and non-reputability of the message. This ensures the transaction data is coming from the genuine node and none of the malicious entities were able to modify the data when communicating over the network. If the hashes match, then the trusted node performs secondary authentication on the transaction by comparing the MACID sent from the edge device. When MACID verification is successful, a random proof-of-authentication nonce which is a random value

Algorithm 2 Transaction Generation in EasyChain

Input: All the edge nodes in the network will have their assigned Private ($P\gamma K_e$) and Public keys(PvK_e).
Output: New block is generated and added to the chain.

for t$_i$ time interval **do**
  Transaction Trx is generated by edge client node (e) including processed information data I_e.
  $Trx \leftarrow createTransaction(I_e)$
  Metadata is added to the transaction Trx
  $Trx \leftarrow Trx.append(Metadata)$
  SHA-256 algorithm is used to compute the hash.
  Digital Signature is generated by using private key of the edge node e.
  $D_{sign} \leftarrow P\gamma K_e(SHA - 256(Trx))$
  MAC address of the edge client node e is appended to the transaction and block is generated.
  Block $B_e \leftarrow Trx^{+}.appendHeader(D_{sign}, MAC)$
  Prepared Block B_e is then published to the entire network
  Generated transaction is then added to the unconfirmed pool before being picked by the trusted node for consensus steps.
  Based on trust value threshold (θ) a trusted node (V) is chosen from the trusted node list $< List > nodes$
  Primary authentication is performed by the chosen trusted node V on digital signature with public key of the source client node.
  $DecryptedMessageHash(MD_{dec}) \leftarrow$ Decrypt(D_{sign},PvK_e)
  $ComputedMessageHash(MD_{com}) \leftarrow SHA - 256(receivedtransaction(Trx^{+}))$
  if $MD_{dec} == MD_{com}$ **then**
    Secondary authentication is performed on the MACID of the transacting node.
    if B_e.MACID == NodeListOfVerifyingNode.getMACID(B_e.SID) **then**
      Random Proof-of-Authentication nonce is generated and appended to the block before broadcating to the network of nodes.
      Confirmed transaction is removed from the unconfirmed pool.
    else
      Ignore the block
      Unauthenticated transaction is removed from the unconfirmed pool.
    end if
  else
    Ignore the block
    Unauthenticated transaction is removed from the unconfirmed pool.
  end if
end for

generated by the trusted node is appended to the block and is published to the entire network. Detailed steps of the generating transaction and creation of blocks are shown in Algorithm 2.

There are multiple types of hashing algorithms, but the most used are Message Digest 5 (MD5), Secure Hashing Algorithm (SHA) 1 and 2, and the SHA-3 candidate called Keccak. SHA-256 produces a 256-bit hash and provides more collision resistance as opposed to MD5 which produces 128-bit output. Even though the performance of SHA-256 is slightly slower compared to MD5, it does not significantly impact the application and provides better security. A comparison of other lightweight hashing functions is done in (Alfrhan et al., 2021) which has shown SHA-256 requires fewer computations compared to keccak and PHOTON hash functions. Hence, SHA-256 is chosen as an optimal choice in the proposed EasyChain application.

Algorithm 3 Proposed Access Control Algorithm for EasyChain

Input: PKI system assigns requester with its own public key $P\upsilon K_d$ and private key $P\gamma K_d$

Requester creates a request transaction along with the timestamp TS at which request is generated
TX_{req}.append(dataRequestInformation,TS)
$\text{Req}_{hash} \leftarrow$ SHA-256(TX_{req})
$\text{DigitalSign}_{requester} \leftarrow \text{Req}_{hash}$.encrypt($P\gamma K_d$)
$\text{TX}^{+}{}_{req} \leftarrow \text{TX}_{req}$.append($\text{DigitalSign}_{requester}$)
Publish the generated request to the network
Requester.publish($\text{TX}^{+}{}_{req}$)
for Every Data Request **do**
    Retrieves public of the requester based on unique identifier assigned
    $P\upsilon K_d \leftarrow$ getPublicKey(requesterID)
    Verify public key against the Access Control List (ACL) at the nodes
    if $P\upsilon K_d$ in ACL **then**
        SHA-256 algorithm is used to compute the hash of the request
        ComputedHash $\leftarrow$ SHA-256(TX_{reqdat})
        Digital sign appended is decrypted using the public key $P\upsilon K_d$ of the requester
        SentHash $\leftarrow \text{DigitalSign}_{requester}$.Decrypt($P\upsilon K_d$)
        Compare the SentHash and ComputedHash
        if ComputedHash == SentHash **then**
            Check the timestamp whether it is within threshold δT
            if $TS \geq TS$-δt OR $TS \leq TS$+δt **then**
                Retrieve requested data from the storage nodes
                $\text{Req}_{data} \leftarrow$ retrieve(TX_{hash})
                Send the retrieved data to the requester
                NetworkNode.publish(Req_{data})
            else
                Discard the request
            end if
        else
            Discard the request
        end if
    else
        Discard the request
    end if
end for

6 THE PROPOSED NOVEL ACCESS CONTROL MECHANISM FOR EASYCHAIN

The proposed PoAh-based EasyChain is designed for private networks in which only the authenticated clients will be able to participate and share the information. It is necessary that other response systems and primary care / Emergency personnel request data from the network. According to HIPPA, healthcare information of individuals should be given the utmost security and privacy. To implement such robust control access methodology, RSA keys are used to identify the requester before any information about the patients is provided. Nodes in the network, along with chain data also maintain an Access Control List (ACL) which will have all the public keys of the requester to which access has been granted. The timestamp of the transaction generated is also appended to the request for avoiding replay attacks. A threshold is defined, and the data request is only processed when a request is reached within the threshold defined. This will make the proposed access control mechanism immune to certain attacks like Replay attacks and Man-in-the-Middle attacks. Detailed steps of data access in the proposed EasyChain are shown in Algorithm 3.

To request data from the private network, the requester node creates a transaction with all the information. A digital signature using the requester's private key is computed and appended to the request transaction before sending it to the private blockchain. Access requests are picked up by one of the network nodes and the public key of the requester is retrieved based on the unique id assigned to the requester. The retrieved public key is then compared with the Access Control List (ACL) implemented at the nodes. Once the requester access has been confirmed, the requester is authenticated based on the digital signature sent to avoid malicious requests from adversaries. If the digital signature is verified, then only the requested data is retrieved and sent back to the requester. In other cases, requests will be discarded thereby providing a robust access control mechanism.

7 THE PROPOSED NOVEL CONSENSUS ALGORITHM - PROOF OF AUTHENTICATION

This section presents PoAh, a novel consensus algorithm proposed for a lightweight blockchain environment for IoT architectures. Unlike traditional consensus algorithms, PoAh validates the devices that are generating the data during the mining process.

All the nodes or participants are connected to the same network and do not have a central entity managing the workflow. All nodes in the network are IoT devices collecting environmental data through sensors. Each node creates transactions with data collected from sensing. Multiple such transactions are collected to form blocks and the block is broadcast to the nodes in the network for the authentication or mining process. The rest of the process is where each consensus algorithm differs and consumes different resources based on the algorithm. Consensus steps for PoW are shown in Figure 4a and Proof-of-Authentication in Figure 4b. From Figure 4a, in the case of PoW all the miners in the network pick transactions from the unconfirmed transaction pool and start the consensus mechanism to find the right nonce. Once one of the competing miner nodes finds the right nonce and publishes a valid block to the network, all the miner nodes will discard their working block and process restarts with a new block made from an updated unconfirmed transaction pool thereby wasting the computational work performed by other miner nodes till then. Along with that, hashcash problem of finding the right nonce is a highly power-consuming step in PoW. On the other hand, the proposed PoAh as shown in Figure 4b picks the trusted node based on trust value which performs the block validation with less resource-intensive digital signature and MACID check. Unlike in PoW, selecting the trusted node based on trust value will also eliminate the wastage of computational work.

Blockchain ledger structure is compared between typical blockchain and the proposed EasyChain is shown in Figure 5. Both transaction and block structures differ from the proposed EasyChain compared to the typical blockchain. EasyChain transaction as shown in Figure 5b has source ID which is a unique ID assigned at the time of client registration into the network, this unique Source ID is used by the trusted node while validating the digital signature of transaction. Along with that, EasyChain is designed for performing data transactions in an IoT environment, and transaction data resides in the corresponding data field in the transaction. Unlike the block structure of PoW as shown in Figure 5a, PoAh doesn't perform the nonce computations, and the fields for the nonce and target difficulty fields are eliminated in EasyChain block structure.

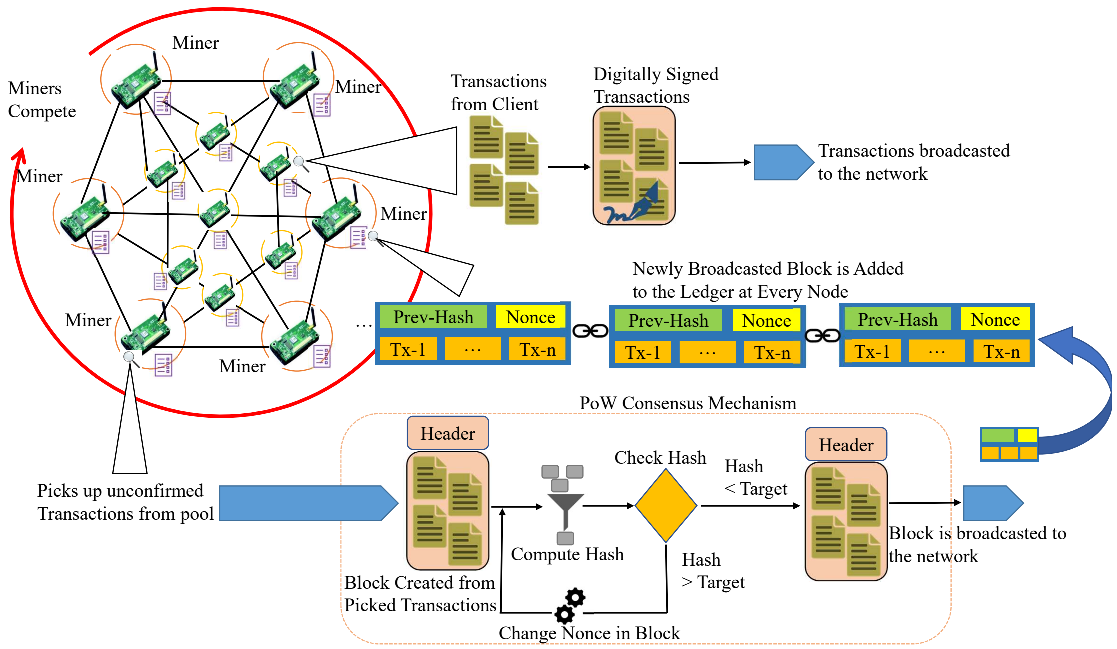


Figure 4a. Proof-of-Work (PoW) consensus algorithm.

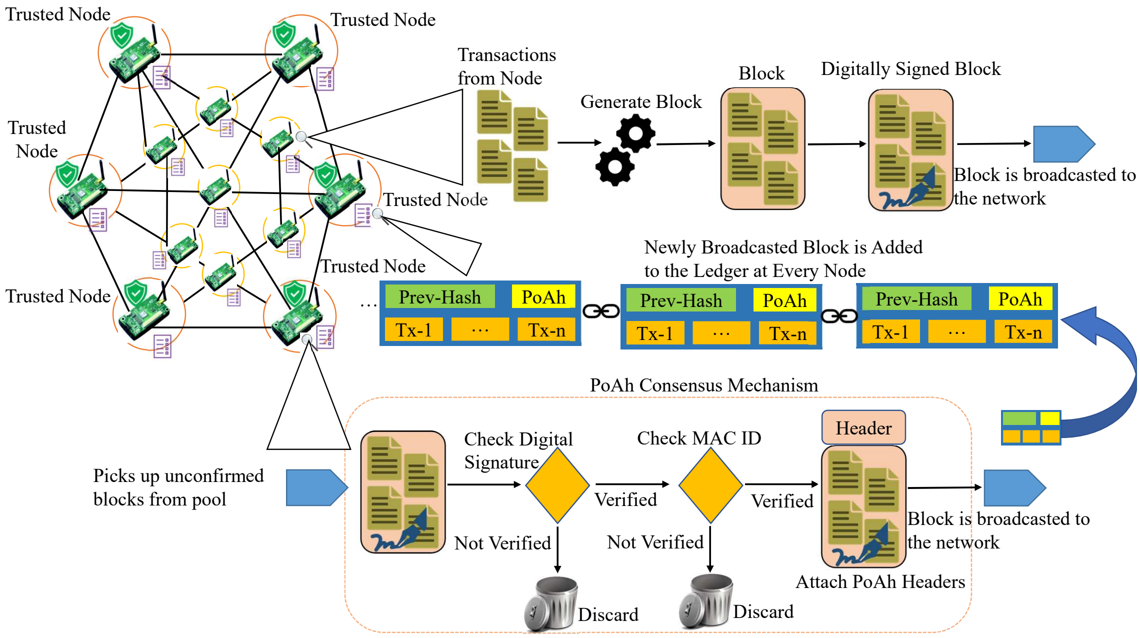


Figure 4b. Proposed Proof-of-Authentication (PoAh) consensus algorithm.

Figure 4. Proof-of-Work (PoW) compared to Proposed Proof-of-Authentication (PoAh)

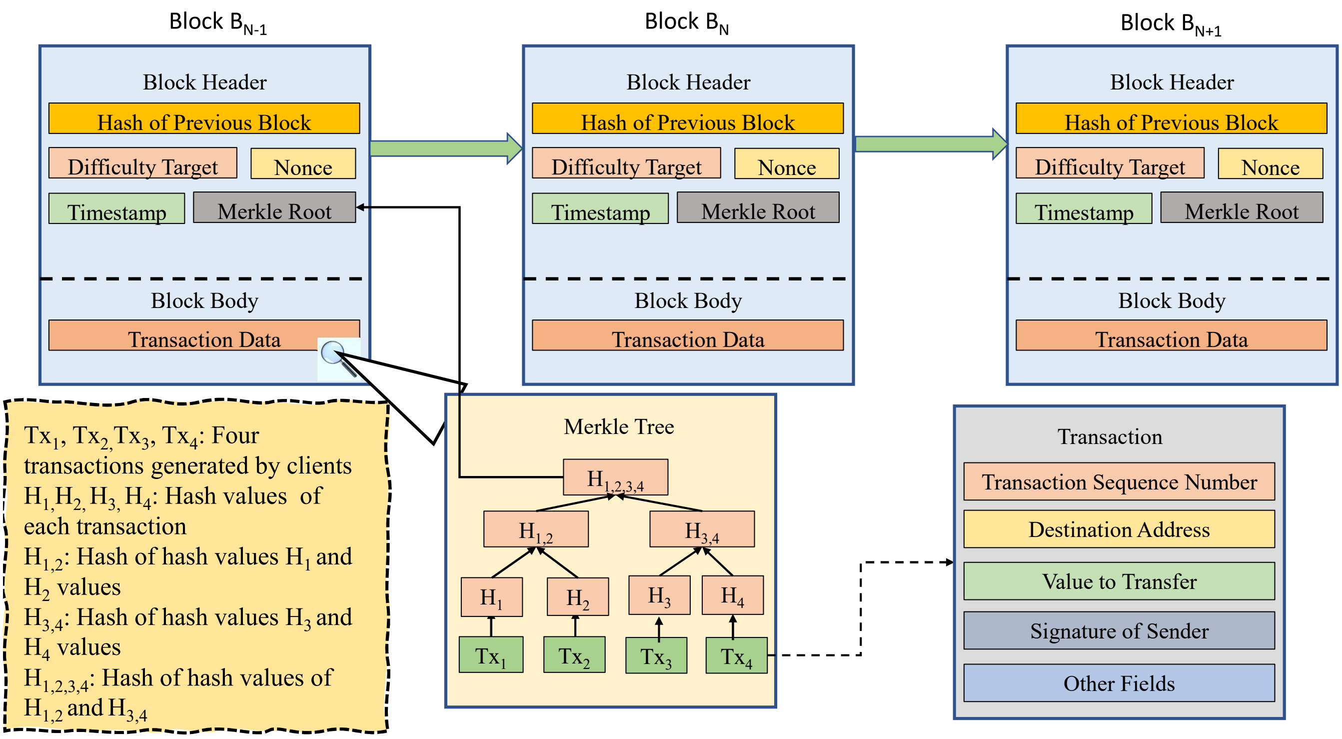


Figure 5a. Block Structure in Typical Blockchain.

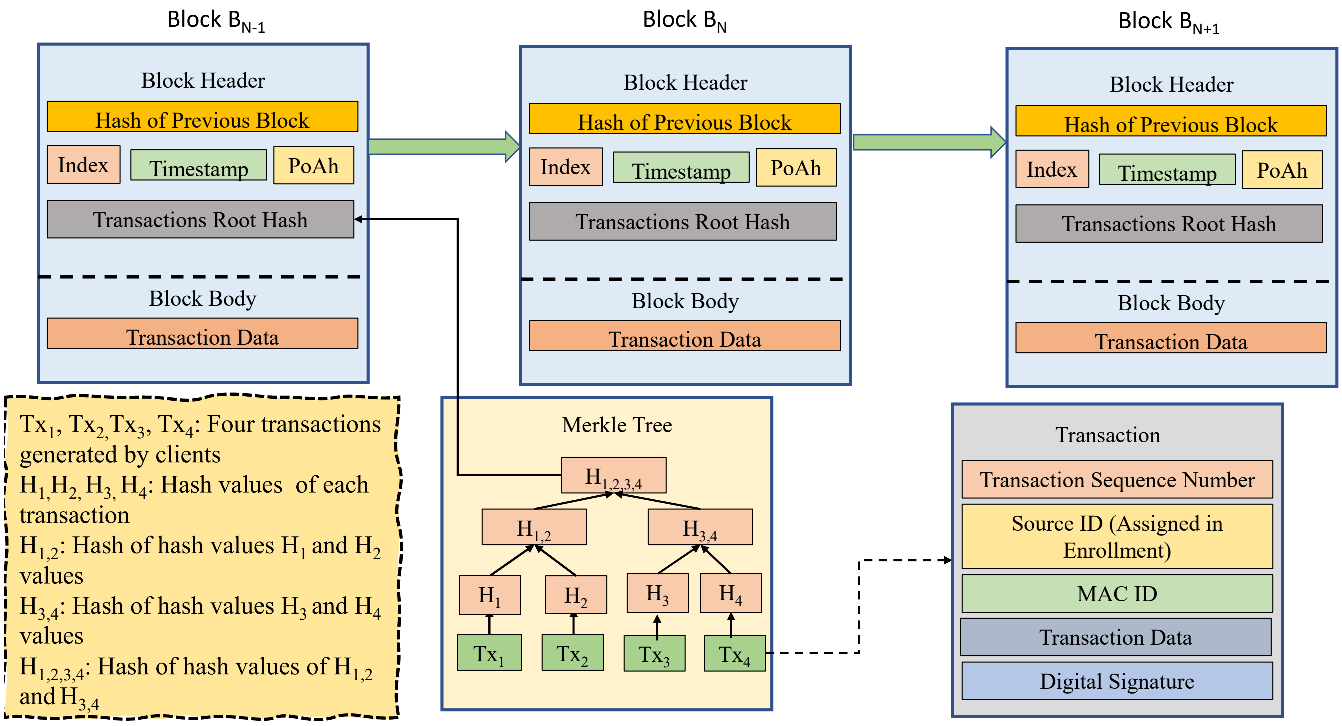


Figure 5b. Block Structure in Proposed EasyChain.

Figure 5. Typical Blockchain Ledger Structure compared to Blockchain Ledger of Proposed EasyChain.

A cryptographic inverse hash is calculated once the transactions are validated in the case of PoW consensus algorithm. Once the calculation is complete, the validated block is broadcast to the network of

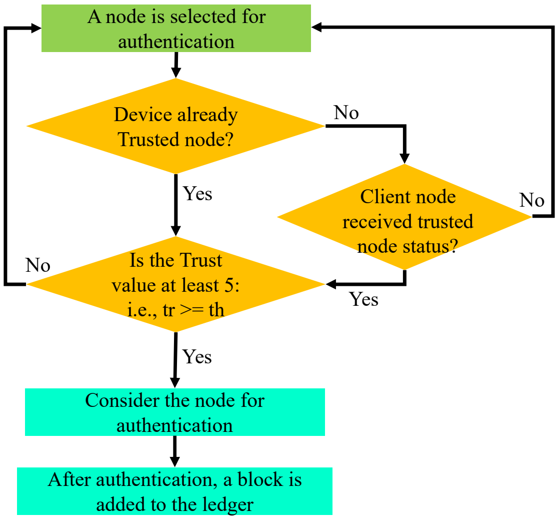


Figure 6. Steps to select authenticated node for PoAh.

devices to add to their local blockchain ledger (Puthal and Mohanty, 2019). In the case of a PoS, a stake is first put by a miner. Based on the stake, the miners are randomly selected to mine the block (Puthal and Mohanty, 2019). Once the block mining is complete, it is broadcast to the network. These processes use high resources, and in some cases, Graphics Processing Units (GPU) for calculating the hash. These high-performance processors are not present on an IoT device.

PoAh is tailored for resource-constrained low-power, low-performance IoT devices. The network is initialized with a limited number of trusted nodes. The trusted nodes are considered secure devices introduced into the network with a trust value higher than zero, “$tr > 0$”. The rest of the devices in the network are client nodes that are assigned a zero-trust value, “tr = 0”. When a block of transactions is authenticated, the trust value is increased by a value of ‘1’, and if a fake block is authenticated, the trust value is decreased by ‘1’. There is a chance for the client nodes to identify the authenticated block to gain trust value. When a client node identifies the block authenticated by a trusted node, the trust value is increased by ‘tr=0.5’. A client node can also identify a fake block which is authenticated by a trusted node to gain a trust value of ‘tr=1’. If the trust value of the trusted node drops below the threshold ‘$tr < th$’, the device can lose its status as a trusted node. A threshold value of ‘5’ is considered in the PoAh implementation and a trust value of ‘10’ is assigned to the trusted nodes. Fig. 6 shows the process of selecting trusted node. Algorithm 4 shows the trust value management in proposed PoAh.

The client node collects the transactions and a source public key to form a block. It is then broadcast across the network. The trusted node receives the block and retrieves the source public key, y for validating the signature on the block. The validation process uses asymmetric cryptography with a public and private key for signature verification. A private key cannot be easily retrieved by the attacker. After the signature is verified, the trusted node evaluates the MAC address for a second round of authentication. Once the block is authenticated by the trusted node, it broadcast the block back to the network by adding a PoAh identifier where others add it to the local blockchain ledgers. Algorithm 5 shows the technical steps of PoAh consensus algorithm.

Algorithm 4 Trust value management in proposed PoAh consensus algorithm.

Input: Initialize the trust value of trusted nodes with a value of 10 and other network nodes with a value of 0.
Output: Updated trust value of the nodes.

1: **for** Selected trusted node N_{sel} with trust value tr_N that is greater than threshold th. **do**
2:   **if** Authenticated block **then**
3:     Authenticated block is broadcast to the network;
4:     **if** Client node N_{client} with trust value tr_{client} finds fake block **then**
5:       $\mathrm{tr}_{client}++$; {Client nodes trust value increases by value 1}
6:       tr_N--; {Trusted node penalized by reducing trust value by 1}
7:       Trusted node status is revoked if new tr_N is less than threshold th;
8:     **else**
9:       **if** Client node N_{client} with trust value tr_{client} performs block validation **then**
10:         $\mathrm{tr}_{client} + 0.5$; {Client nodes trust value increases by 0.5}
11:         tr_N++; {Selected trusted node trust value increases by 1}
12:       **else**
13:         tr_N++; {Only selected trusted node trust value increases by 1}
14:       **end if**
15:     **end if**
16:   **else**
17:     tr_N--; {Selected trusted node is not available}
18:     Trusted node status is revoked if new tr_N is less than threshold th;
19:   **end if**
20:   Select new trusted node and GOTO $(Step - 1)$;
21: **end for**

Algorithm 5 Procedure of PoAh consensus algorithm.

Input: $SHA-256$ hash is used at all nodes. Every participant has private (PrK) and public keys (PuK).
Output: Authenticated Blocks that are added to the ledger.

1: $(Trx^+) \rightarrow$ blocks; {Multiple transactions are combined to form blocks.}
2: (S_{PrK})(block) $\rightarrow$ broadcast; {Block is signed with private key and broadcast to the network.}
3: (V_{PuK})(block) $\rightarrow$ MAC Checking; {Trsuted nodes authenticates the block with source public key}
4: **if** $Authenticated$ **then**
5:   $block||PoAh(D) \rightarrow$ broadcast; {Authenticated block is broadcast to network with trusted node signature}
6:   $H(block) \rightarrow$ Add blocks into chain; {If block has trusted node signature, they add to block.}
7: **else**
8:   DROP the block; {If block is not authentic, it is discarded.}
9: **end if**
10: GOTO $(Step - 1)$ for next block;

8 EXPERIMENTAL EVALUATIONS

This section presents the simulation results of a large-scale study and a test-bench was designed for small-scale experimental results of the proposed Blockchain.

8.1 Simulation Evaluation

The proposed EasyChain is implemented using the Python programming language. An IoT System with 4 nodes among which one node has been given a trust value greater than the threshold value to act as a validating node. For experimental setup, all nodes are implemented using the Raspberry Pi 4 Model B which is based on the Broadcom BCM2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC at 1.8GHz with 4GB LPDDR4-3200 SDRAM. To quantify the computational capabilities of the node, OpenSSL is used

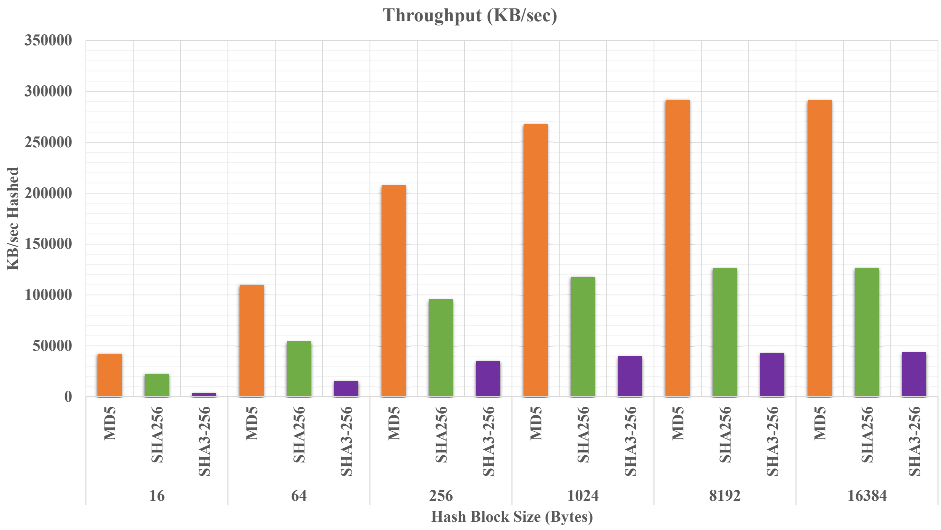
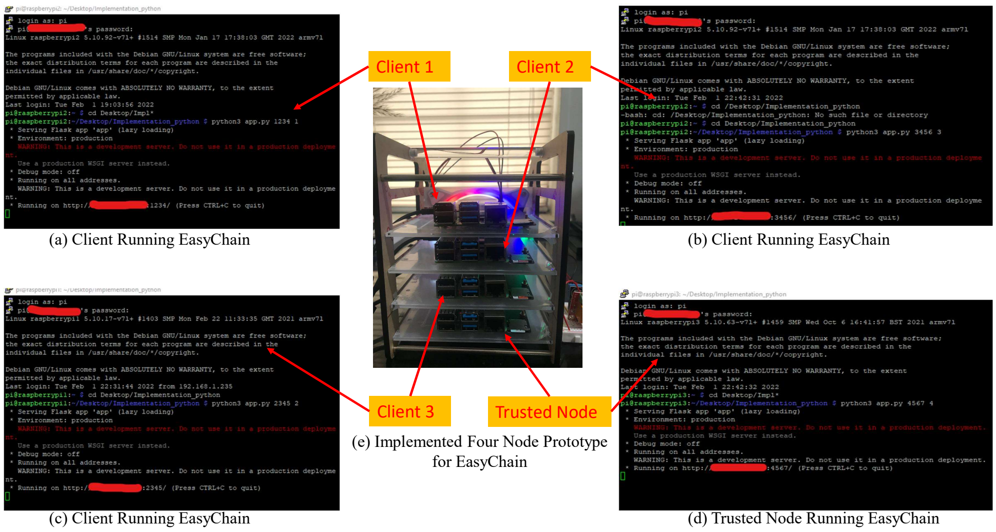


Figure 7. Cryptographic Digest Performance of Node with Varied Block Size and Digest Algorithm.



(a) Client Running EasyChain (b) Client Running EasyChain (c) Client Running EasyChain (d) Trusted Node Running EasyChain (e) Implemented Four Node Prototype for EasyChain

Figure 8. Proposed EasyChain Experimental Setup.

to perform benchmark tests to measure node cryptographic performance. A set of digest algorithms are selected for testing which includes MD5, SHA-256, and SHA3-256. Throughput results from the benchmark test can be seen in Figure 7. Experimental setup for implemented EasyChain is shown in Figure 8. As the data size used for simulation evaluation is small, one of the nodes with a 32GB SD Card acts as a storage node in the current experimental setup. If large amounts of storage are needed in real-time applications, an SSD can be interfaced with the Raspberry Pi 4 node through USB 3.0 port. Off-chain storage using Inter-planetary File System (IPFS) can also be implemented as a solution for data storage. RSA public cryptography system is used in EasyChain for encryption, digital signatures, and verifying the signatures. Block format for implemented EasyChain follows $< SourceID, DigitalSignature, Tx1, Tx2, ... >$.

Fig. 9 shows the ledger structure along with other chain information. Blockchain ledger consists of the mined blocks which are added to the chain along with pending transactions, registered network nodes, and their corresponding information like MAC address for PoAh secondary authentication.

Figure 9. Ledger Structure Showing Genesis Block for Implemented EasyChain.

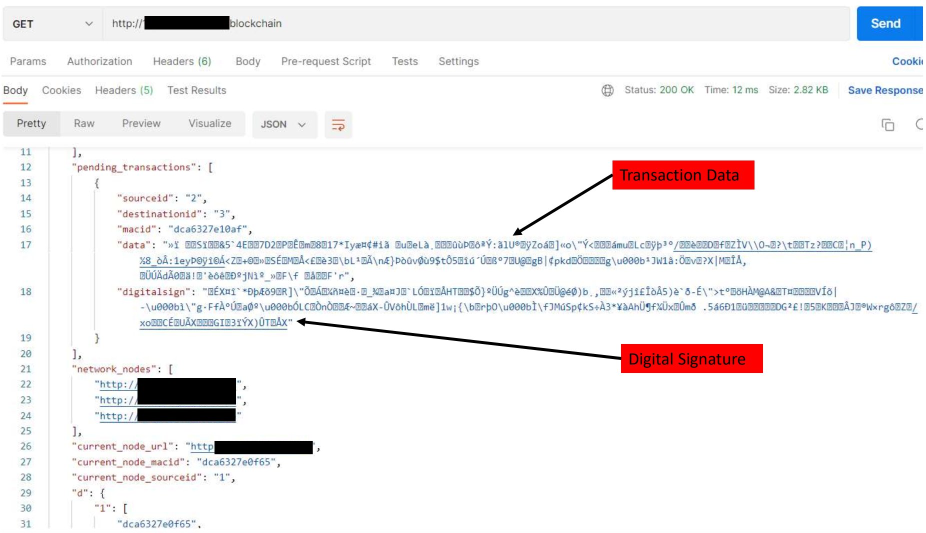


Figure 10. Transaction Added to Unconfirmed Transaction Pool in EasyChain.

Sample monitoring data which consists of essential information like patient id, Body Temperature, Respiratory Rate, Saturated Oxygen level (SpO2), and Blood Pressure is used for performing the transactions from the client node. Before sending the patient data, the transaction is signed by the private key and the broadcast transaction will be added to the unconfirmed transaction pool at each network node. Added unconfirmed transaction can be seen in Fig. 10.

One of the trusted nodes in the network will pick up the transactions from the unconfirmed transaction pool and perform PoAh consensus. Once the consensus is reached, it will be added as a new block in the chain at every peer node and the corresponding transaction will be purged from the unconfirmed pool. The confirmed block is shown in Fig. 11.

8.2 Performance Evaluation

Transaction time and block generation times are analyzed to evaluate the performance of implemented EasyChain. Timestamps are generated at multiple checkpoints of block processing to record the time taken

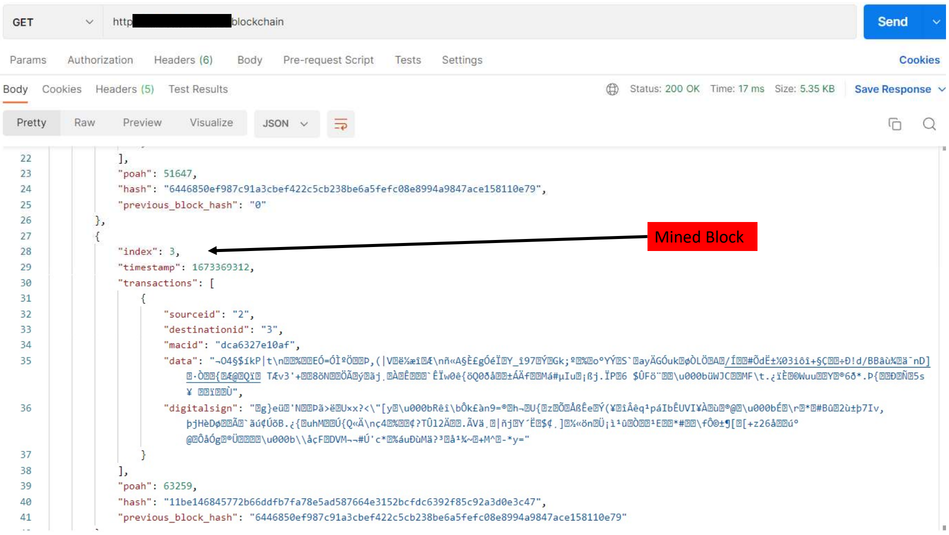


Figure 11. Block Added to Chain after Performing Proposed PoAh Consensus by Trusted Node.

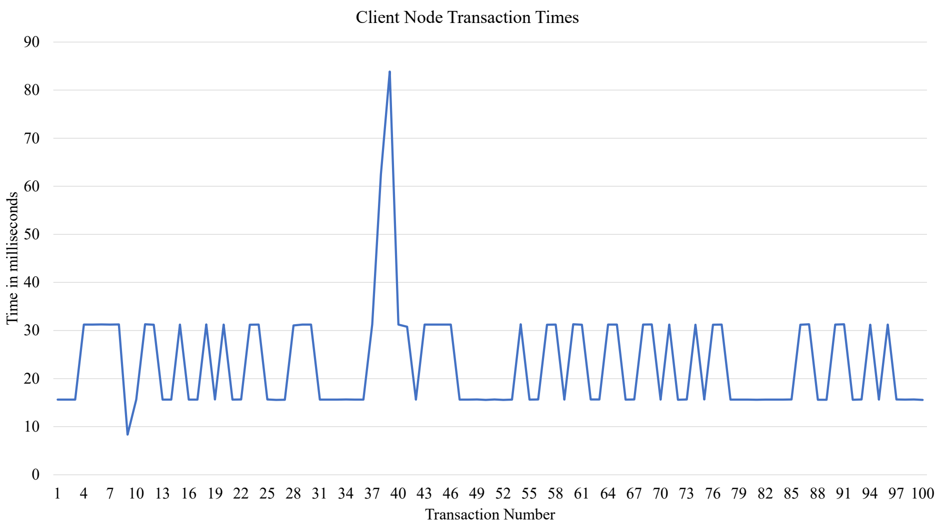


Figure 12. Time Taken for Client Node to Send Transaction to Trusted Node.

for the transaction to reach the trusted node and the time taken by the trusted node to perform the consensus mechanism and add a new block.

Timestamp t_{cp} is the time at which the client node has collected the data from the sensing elements and prepares a transaction whereas timestamp t_{tr} is the time taken for the client transaction to reach the trusted node. Client transaction time δ_{ct} is computed from these timestamps.

$$\delta_{ct} = t_{tr} - t_{cp} \tag{1}$$

100 transactions are performed from a client node in the implemented EasyChain and measured transaction times can be seen in Fig. 12.

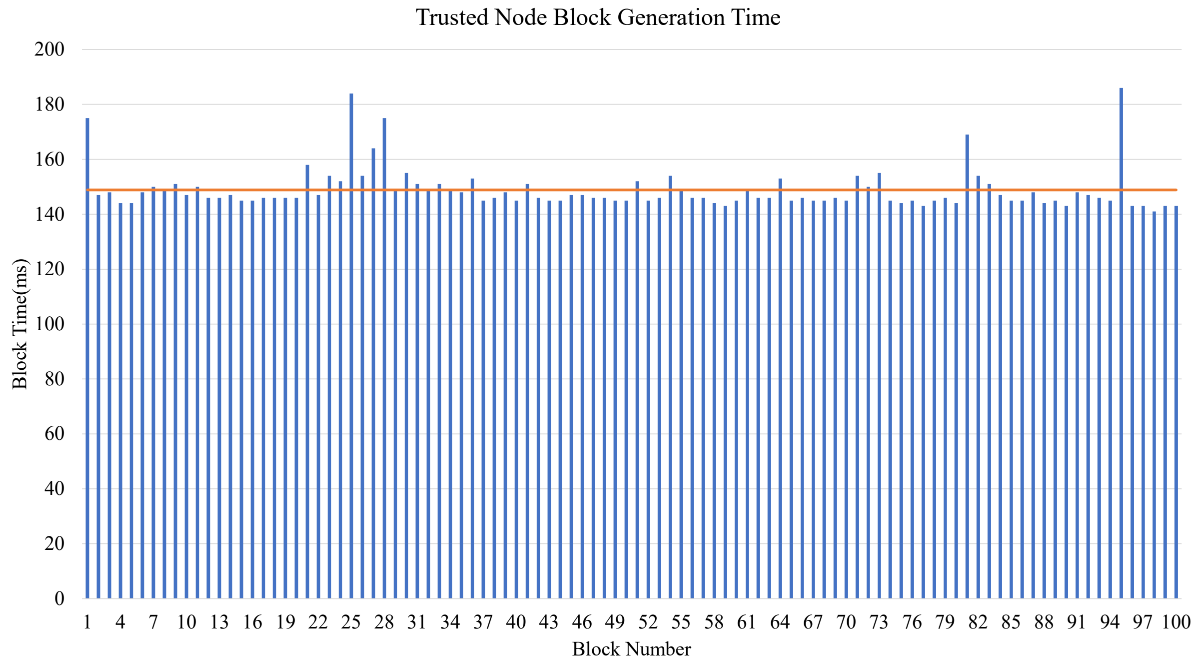


Figure 13. Time Taken for Trusted Node to Generate a New Block.

Table 3. Transaction and Block Time in Implemented EasyChain.

	Client Node	Trusted Node
Minimum Time (ms)	8.34	141
Maximum Time (ms)	83.87	186
Average Time (ms)	23.09	148.89

Similarly, block generation time is measured from the timestamps recorded t_{tr} being time recorded when a transaction reached the trusted node and t_{tm} being the time at which the block is mined after performing PoAh consensus.

$$\delta_{tb} = t_{tm} - t_{tr} \quad (2)$$

Computed block generation times can be seen in Fig. 13. Minimum, Maximum, and Average times are computed and are shown in Table 3. We can see the minimum, maximum, and average transaction times for the client node are 8.34ms, 83.87ms, and 23.09ms respectively. Similarly, the minimum, maximum, and average block times of trusted nodes are 141ms, 186ms, and 148.9ms respectively.

8.3 Power Consumption

Another challenge for integrating blockchain into a resource-constrained IoT environment is power consumption. Implemented test-bed is evaluated for power consumption by using an electrical meter connected to the power outlet as shown in Fig. 14. Power is measured when both implemented systems are in an ideal state and when SBC is processing the data. Power consumed by the client node, trusted node, and storage nodes in both scenarios is shown in Table 4.

Figure 14. Electric Meter Setup for Measuring Power Consumption in Implemented EasyChain

Table 4. Power Consumption of Different Nodes in Implemented EasyChain.

	Client Node	Trusted Node	Storage Node
Max Power Consumption in Watts	1.8	2.5	3.6
Min Power Consumption in Watts	1.5	2	3.1

Power consumption of the client node is minimum at 1.5 Watts when SBC is in an idle state whereas it is maximum 1.8 Watts when collecting the information and performing the transaction. Similarly, the trusted node also consumed lower power of 2 Watts at idle state and 2.5 Watts when performing the consensus mechanism for the received transaction. Storage node consumes higher power compared to the other two types of nodes with power ranging from 3.1 Watts to 3.6 Watts. Power consumption is shown in Fig. 15. Comparison of proposed Proof-of-Authentication (PoAh) with some of the established protocols can be seen in Table 2.

9 DISCUSSION ON PROPOSED PROOF OF AUTHENTICATION CONSENSUS PROTOCOL

PoAh consensus algorithm authenticates the devices that are transmitting the data in contrast to other consensus algorithms such as PoW and PoS which validate only the transactions sent by the nodes. PoAh uses significantly less energy and resources which is suitable for resource-constrained IoT environments. In PoAh, the block has the patient data collected by the sensors, the identity of the device on the patient, and the timestamp when the block is generated. All the nodes are connected to the same network through a wired or wireless interface using IPv4. The MAC address is used as the identification for the devices during the block generation. Once the block is validated by the trusted nodes, it is broadcast to the network with the signature of the trusted node and other nodes add to their local blockchain ledger. The following claims are made in the paper to validate PoAh is scalable and suited for the IoE.

Claim – 1: Block validation in PoAh uses less resources.

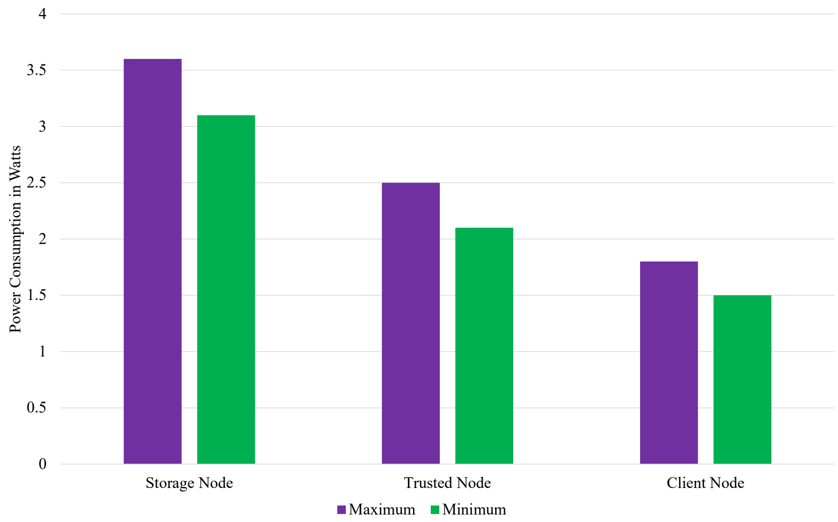


Figure 15. Power Consumption of Different Nodes in EasyChain

Discussion: In the consensus algorithms such as PoW, to validate the transactions, the inverse hash of the block is calculated by the miners. This calculation is a resource-heavy process, which utilizes equivalent energy consumed by two households in a day (Zyskind et al., 2015). IoT environment has low-power low-performance devices that cannot perform such computationally intensive tasks. PoAh uses a device authentication mechanism to validate the nodes transmitting the data. Validating a signature consumes significantly less power and requires fewer resources compared to the calculation of inverse hash.

Claim – 2: Time taken to authenticate devices in PoAh is less without compromising security.

Discussion: In PoW, block validation takes 10 minutes and a new block is generated after that (Zyskind et al., 2015). In any IoE application, data collection and transmission cannot afford to spend 10 minutes for a new block generation. IoT devices are used to monitor the source at regular intervals. Device authentication in PoAh takes minimal time. Experimental evaluations show PoAh is 1,000 times faster than PoW (Dorri et al., 2017).

Claim – 3: A substantial blockchain based security is provided by PoAh.

Discussion: IoT applications deal with devices that send data in real time. So, a security primitive tailored for such an application is necessary. A cryptographic solution is a sufficient protection in the current proposed scenario, unlike the cryptocurrencies (Puthal et al., 2018). PoAh integrates the cryptographic security provided by PoW ignoring the block evaluation of computing the inverse of the hash. The issues in PoW, unstable network connectivity, and 51 % attack are addressed in the proposed consensus algorithm. All devices in the network are capable of data generation and trusted nodes authenticate the blocks and trusted peers (solves the 51% attack issue) can authenticate and add blocks into the chain.

Claim – 4: PoAh is a better consensus algorithm for IoT integration compared to the existing algorithms.

Discussion: A consensus algorithm is responsible for taking the decision to validate and add a block to the Blockchain ledger. Widely used consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), and Proof of Authority (PoAu) are resource hungry and consume more power (Andoni et al., 2019). Block mining takes around 10 minutes in the case of PoW and around 1 hour to get accepted to the ledger. PoAh addresses such issues in the IoT architectures.

10 CONCLUSIONS

This paper provides EasyChain, a novel PoAh-based blockchain for the IoE. The proposed blockchain does not have a centralized entity by building a lightweight security solution using PoAh. EasyChain is validated using theoretical analysis, simulation, and a real-time experimental evaluation. The results show a promising IoE integration of blockchain. The proposed algorithm can be deployed across multiple devices and environments, when a patient is present in the hospital, at home, or in an ambulance as EasyChain does not rely on a certain communication protocol.

As a future work, the framework can be extended to add multiple layers of security, adding a hardware-assisted security module like Physical Unclonable Function modules to the proposed work. A user-friendly GUI along with the blockchain explorer to check the status of the blockchain and retrieve transactions easily will be implemented. Business logic implementation is difficult in the current EasyChain implementation as it doesn't support smart contracts and to implement any business logic, the base code must be modified. Easier integration of business logic will be the next step we will work on to further improve our proposed EasyChain architecture.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that they have no conflict of interest and there was no human or animal testing, or participation involved in this research. All data were obtained from public domain sources.

ACKNOWLEDGMENTS

A preliminary version of this study has been presented at ICCE 2019 (Puthal et al., 2019). An extended version of this work has been archived at (Puthal et al., 2020).

REFERENCES

Alfandi, O., Khanji, S., Ahmad, L., and Khattak, A. (2020). A survey on boosting IoT security and privacy through blockchain. *Cluster Computing* 24, 37–55. doi:10.1007/s10586-020-03137-8

Alfrhan, A., Moulahi, T., and Alabdulatif, A. (2021). Comparative study on hash functions for lightweight blockchain in internet of things (IoT). *Blockchain: Research and Applications* 2, 100036. doi:10.1016/j.bcra.2021.100036

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., et al. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews* 100, 143–174

[Dataset] Back, A. (2002). Hashcash - a denial of service counter-measure. Last accessed on 2023-01-25

Castanho, M. S., Ferreira, F. A. F., Carayannis, E. G., and Ferreira, J. J. M. (2019). SMART-C: Developing a "Smart City" Assessment System Using Cognitive Mapping and the Choquet Integral. *IEEE Transactions on Engineering Management* , 1–12doi:10.1109/TEM.2019.2909668

Castro, M. (1999). Practical byzantine fault tolerance. In *USENIX Symposium on Operating Systems Design and Implementation*

[Dataset] Chase, B. and MacBrough, E. (2018). Analysis of the xrp ledger consensus protocol. doi:10.48550/ARXIV.1802.07242

Corbett, J., Wardle, K., and Chen, C. (2018). Toward a sustainable modern electricity grid: The effects of smart metering and program investments on demand-side management performance in the US electricity sector 2009-2012. *IEEE Transactions on Engineering Management* 65, 252–263. doi:10.1109/TEM.2017.2785315

Dorri, A., Kanhere, S. S., Jurdak, R., and Gauravaram, P. (2017). Blockchain for IoT Security and Privacy: The Case Study of a Smart Home. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 618–623

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., and Sikdar, B. (2019). A survey on iot security: Application areas, security threats, and solution architectures. *IEEE Access* 7, 82721–82743. doi:10.1109/ACCESS.2019.2924045

Huang, J., Kong, L., Chen, G., Wu, M.-Y., Liu, X., and Zeng, P. (2019). Towards secure industrial IoT: Blockchain system with credit-based consensus mechanism. *IEEE Transactions on Industrial Informatics* 15, 3680–3689. doi:10.1109/tii.2019.2903342

Huang, Z., Su, X., Zhang, Y., Shi, C., Zhang, H., and Xie, L. (2017). A Decentralized Solution for IoT Data Trusted Exchange Based-on Blockchain. In *Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC)*. 1180–1184

[Dataset] King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Last accessed on 2023-01-25

Kshetri, N. (2017). Can Blockchain Strengthen The Internet of Things? *IT Professional* 19, 68–72

Kumar, A., Krishnamurthi, R., Nayyar, A., Sharma, K., Grover, V., and Hossain, E. (2020). A novel smart healthcare design, simulation, and implementation using healthcare 4.0 processes. *IEEE Access* 8, 118433–118471. doi:10.1109/ACCESS.2020.3004790

Kuzmin, A. (2017). Blockchain-Based Structures for a Secure and Operate IoT. In *Proceedings of the Internet of Things Business Models, Users, and Networks*. 1–7

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Transactions on Programming Languages and Systems* 4, 382–401. doi:10.1145/357172.357176

Li, K., Li, H., Hou, H., Li, K., and Chen, Y. (2017). Proof of Vote: A High-Performance Consensus Protocol Based on Vote Mechanism & Consortium Blockchain. In *Proceedings of the IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. 466–473. doi:10.1109/HPCC-SmartCity-DSS.2017.61

Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., and Imran, M. A. (2021). A scalable multi-layer PBFT consensus for blockchain. *IEEE Transactions on Parallel and Distributed Systems* 32, 1146–1160. doi:10.1109/tpds.2020.3042392

[Dataset] Mazieres, D. (2015). The stellar consensus protocol:a federated model for internet-level consensus. Last accessed on 2023-01-25

Misra, S., Mukherjee, A., Roy, A., Saurabh, N., Rahulamathavan, Y., and Rajarajan, M. (2021). Blockchain at the Edge: Performance of Resource-Constrained IoT Networks. *IEEE Transactions on Parallel and Distributed Systems* 32, 174–183. doi:10.1109/TPDS.2020.3013892

[Dataset] Mitra, A., Vangipuram, S. L. T., Bapatla, A. K., Bathalapalli, V. K. V. V., Mohanty, S. P., Kougianos, E., et al. (2022). Everything you wanted to know about smart agriculture. doi:10.48550/ARXIV.2201.04754

Mohanty, J., Mishra, S., Patra, S., Pati, B., and Panigrahi, C. R. (2020). IoT security, challenges, and solutions: A review. In *Advances in Intelligent Systems and Computing* (Springer Singapore). 493–504. doi:10.1007/978-981-15-6353-9_46

Moreno, M. V., Terroso-Saenz, F., Gonzalez-Vidal, A., Valdes-Vela, M., Skarmeta, A. F., Zamora, M. A., et al. (2017). Applicability of Big Data Techniques to Smart Cities Deployments. *IEEE Transactions on Industrial Informatics* 13, 800–809. doi:10.1109/TII.2016.2605581

Mukhopadhyay, S. C., Tyagi, S. K. S., Suryadevara, N. K., Piuri, V., Scotti, F., and Zeadally, S. (2021). Artificial intelligence-based sensors for next generation IoT applications: A review. *IEEE Sensors Journal* 21, 24920–24932. doi:10.1109/jsen.2021.3055618

Nayak, A. and Dutta, K. (2017). Blockchain: The Perfect Data Protection Tool. In *Proceedings of the International Conference on Intelligent Computing and Control (I2C2)*. 1–3

[Dataset] NemProject (2018). Nem technical reference. Last accessed on 2023-01-25

Novo, O. (2018). Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal* 5, 1184–1195

[Dataset] Olson, K., Bowman, M., Mitchell, J., Amundson, S., Middleton, D., and Montgomery, C. (2018). Sawtooth: An introduction. Last accessed on 2023-01-25

Ouaddah, A., Abou Elkalam, A., and Ait Ouahman, A. (2016). FairAccess: A New Blockchain-Based access Control Framework for the Internet of Things. *Security and Communication Networks* 9, 5943–5964

Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E., and Yang, C. (2018). The Blockchain as a Decentralized Security Framework. *IEEE Consumer Electronics Magazine* 7, 18–21

Puthal, D. and Mohanty, S. P. (2019). Proof of Authentication: IoT-Friendly Blockchains. *IEEE Potentials Magazine* 38, 26–29

Puthal, D., Mohanty, S. P., Nanda, P., Kougianos, E., and Das, G. (2019). Proof-of-Authentication for Scalable Blockchain in Resource-Constrained Distributed Systems. In *Proceedings of the 37th IEEE International Conference on Consumer Electronics (ICCE)*

Puthal, D., Mohanty, S. P., Yanambaka, V. P., and Kougianos, E. (2020). PoAh: A Novel Consensus Algorithm for Fast Scalable Private Blockchain for Large-scale IoT Frameworks. *arXiv Computer Science* abs/2001.07297

Puthal, D., Nepal, S., Ranjan, R., and Chen, J. (2016). Threats to networking cloud and edge datacenters in the internet of things. *IEEE Cloud Computing* 3, 64–71

Puthal, D., Nepal, S., Ranjan, R., and Chen, J. (2017). DLSeF: A Dynamic Key-Length-Based Efficient Real-Time Security Verification Model for Big Data Stream. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 51

Qu, Y., Pokhrel, S. R., Garg, S., Gao, L., and Xiang, Y. (2021). A Blockchained Federated Learning Framework for Cognitive Computing in Industry 4.0 Networks. *IEEE Transactions on Industrial Informatics* 17, 2964–2973. doi:10.1109/TII.2020.3007817

Shahzad, A. and Kim, K. (2019). FallDroid: An Automated Smart-Phone-Based Fall Detection System Using Multiple Kernel Learning. *IEEE Transactions on Industrial Informatics* 15, 35–44. doi:10.1109/TII.2018.2839749

S. P. Mohanty, Yanambaka, V. P., Kougianos, E., and Puthal, D. (2020). PUFchain: Hardware-Assisted Blockchain for Sustainable Simultaneous Device and Data Security in Internet of Everything (IoE). *IEEE Consumer Electronics Magazine* 9, 8–16

Wang, E. K., Liang, Z., Chen, C.-M., Kumari, S., and Khan, M. K. (2020). PoRX: A reputation incentive scheme for blockchain consensus of IIoT. *Future Generation Computer Systems* 102, 140–151. doi:10.1016/j.future.2019.08.005

Wang, Y. and Malluhi, Q. M. (2019). The Limit of Blockchains: Infeasibility of a Smart Obama-Trump Contract. *Communications of the ACM* 62, 64–69

Xin, W., Zhang, T., Hu, C., Tang, C., Liu, C., and Chen, Z. (2017). On Scaling and Accelerating Decentralized Private Blockchains. In *Proceedings of the IEEE 3rd International Conference on Big Data Security on Cloud (bigdatasecurity), IEEE international conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. 267–271

Xu, J., Gu, B., and Tian, G. (2022). Review of agricultural IoT technology. *Artificial Intelligence in Agriculture* 6, 10–22. doi:10.1016/j.aiia.2022.01.001

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things journal* 1, 22–32

Zhaofeng, M., Xiaochang, W., Jain, D. K., Khan, H., Hongmin, G., and Zhen, W. (2020). A Blockchain-Based Trusted Data Management Scheme in Edge Computing. *IEEE Transactions on Industrial Informatics* 16, 2013–2021. doi:10.1109/TII.2019.2933482

Zou, J., Ye, B., Qu, L., Wang, Y., Orgun, M. A., and Li, L. (2018). A Proof-of-Trust Consensus Protocol for Enhancing Accountability in Crowdsourcing Services. *IEEE Transactions on Services Computing* , 1–14doi:10.1109/TSC.2018.2823705

Zyskind, G., Nathan, O., and Pentland, A. S. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *Proceedings of the IEEE Security and Privacy Workshops* (IEEE), 180–184