Journal of Information Systems Engineering and Management
2025, 10(17s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Scalable Approach to IoT Data Retrieval Using Blockchain, Aggregate Signatures, and Bloom Filters

Shweta Babu Prasad[1], Ashok Kumar A R[2], Rajini V Honnungar[3]

[1]*Department of CSE, R V College of Engineering, Bengaluru, India*
[2]*Department of CSE, R V College of Engineering, Bengaluru, India*
[3]*Department of ECE, RNS Institute of Technology, Bengaluru, India*
[1]*shwetababup@rvce.edu.in*

ARTICLE INFO

Received: 09 Dec 2024

Revised: 30 Jan 2025

Accepted: 10 Feb 2025

ABSTRACT

The Internet of Things (IoT) is a network of a variety of devices that generate diverse data and need scalable, efficient, and secure management. Traditional, centralized solutions often fail to meet the security and scalability needs of such networks. Blockchain enhances data security and integrity due to its decentralized nature. However, IoT devices are resource constrained. To address this, we propose a multi-layer blockchain framework. It utilizes several permissioned blockchains for parallel processing, which enhances scalability and access control. The framework works with decentralized storage like the InterPlanetary File System (IPFS). Bloom filters optimize data retrieval by filtering out non-existent content. Additionally, we incorporate the High-Performance Edwards Curve Aggregate Signature (HECAS). HECAS boosts transaction speed and block validation by 10%. It also cuts storage costs by 40%. The system ensures signatures can't be denied and verifies them quickly. This solves key blockchain issues in IoT. Tests show data detection is fast using Bloom's filter and takes 1.21 seconds for present data and 0.02 seconds to notify the absence of data.

Keywords: Aggregate signature, digital signature, elliptic curve cryptography, HECAS, IPFS

1. INTRODUCTION

The blockchain is the Bitcoin technology's backbone network for processing transactions. Blockchains are decentralized and immutable. Across numerous industries, blockchain is revolutionizing finance, healthcare, agriculture, and smart Internet of Things (IoT) applications. Satoshi Nakamoto's groundbreaking work [1], paved the way for these advancements. IoT networks are now widely used across various applications in business, healthcare, security, tracking, smart homes, and smart grids, to name a few. This can be attributed to the accessibility of IoT devices with limited resources that are economical and practical [2,3]. IoT frameworks are frequently created with centralized technologies and structure. This implies a centralized server collects and processes data from IoT devices. However, this approach makes IoT networks vulnerable to privacy and security issues resulting from both physical and cyberattacks [4,5]. Due to its inherent characteristics, blockchain technology emerges as a robust solution for secure IoT network implementations [6]. Blockchain-powered IoT networks offer a reliable and efficient way to connect and exchange data between physical and virtual devices equipped with sensors and actuators over the internet [7].

Blockchain, a particular type of distributed ledger, tracks, supervises, and manages IoT devices using cryptography and decentralization [8,9]. Blockchain transactions are highly reliable as they eliminate the need for intermediaries. Its intrinsic qualities—immutability, accountability, and decentralization—provide significant benefits like enhanced security, data protection against unauthorized access, and full process traceability. By integrating these advanced technologies, secure and scalable IoT networks can be developed, enabling seamless data sharing among stakeholders, systems, and devices.

While blockchain provides strong access control, managing the vast amounts of data generated by IoT devices poses a significant challenge. The Inter Planetary File System (IPFS), a peer-to-peer distributed file system, has been

suggested as an excellent solution for decentralized data storage in IoT networks. IPFS facilitates efficient and secure storage by distributing data across multiple nodes, eliminating the need for centralized data servers and reducing the risk of data loss [10]. However, the decentralized nature of IPFS also brings complexities in data retrieval, especially in large-scale networks where data may be spread across numerous nodes.

To overcome these challenges, integrating Bloom filters—a space-efficient probabilistic data structure—into the IPFS framework can significantly improve data retrieval efficiency [11]. Bloom filters enable quick and efficient existence checks of data within the network, greatly reducing the time and computational resources needed to find and retrieve specific pieces of data. This is especially critical in IoT environments, where rapid access to data is vital for real-time decision-making and operations.

Various digital signature systems, such as aggregate, blind, group, proxy, and ring, are detailed in [12]. An aggregate signature can combine n messages from n nodes into one fixed-length signature. This single signature can provide non-repudiation for all n messages across n nodes. Aggregate signatures not only reduce storage requirements but also lower the computational and communication costs associated with verification. As noted in [13], this type of signature is ideal for tasks with limited storage, network, and computing resources. Additionally, [14] demonstrates that digital signature techniques ensure information nonrepudiation, blockchain integrity, authenticity, and identity verification.

2. RELATED WORK

The integration of blockchain, IPFS, and Bloom filters in IoT networks represents a pioneering research area that tackles the crucial issues of security, scalability, and efficient data retrieval in distributed environments. This literature survey highlights the most recent advancements and contributions in these fields.

The decentralized nature of blockchain helps counter security threats such as unauthorized access and data tampering, boosting the reliability of IoT systems [15]. Spatial blockchain [16] further optimizes this by segmenting blockchains into spatial units and consolidating data over time, thereby reducing the overall size of the blockchain. RapidChain [17] significantly boosts throughput by applying sharding to public blockchains and can handle Byzantine faults impacting nearly one-third of participants. Each shard is formed by randomly assigning participants to committees, ensuring that compromised members in any committee do not exceed one-half. Afterward, each committee uses the synchronous Byzantine consensus method to reach an agreement.

[18] introduces a streamlined hierarchical access control system built on blockchain and multi-chaincode for IoT networks, using a clustering strategy with blockchain managers to enhance scalability. Elastico [19] organizes miners into several groups, allowing transaction throughput to rise linearly with the total processing power dedicated to mining. By increasing the number of nodes from 100 to 1600, the number of blocks generated per epoch can rise from 1 to 16, with the epoch period increasing from 600 to 711 seconds as a result. Omniledger [20], leveraging two PoS blockchain technologies known as Ouroboros and Algorand, enhances blockchain scalability and bias-resistant validators through parallel transaction processing in an intra-shard manner. RandHound [21] manages the secondary key security channel. A checkpoint-based method is used to improve efficiency, enabling miners to create new blocks without downloading the entire blockchain history. This approach achieves a throughput of 13,000 transactions per second (TPS), even with an adversary rate of 12.5% and 1,800 hosts distributed across 25 shards. However, the system requires a considerable amount of time (in the order of minutes) to bootstrap each epoch.

According to [22], the elliptic curve digital signature algorithm (ECDSA)-based modern cryptography has gained popularity recently due to its performance, privacy, and the efficient length of keys and signatures. An elliptic curve (EC) is an effective algorithm that utilizes minimal resources and involves few mathematical operations. This cubic curve is represented by Equation (1).

$$E: y^2 \equiv x^3 + a.x + b \bmod p \quad (1)$$

The Edwards curve, as a generalization of elliptic curves, is detailed in [23] to provide robust security on devices with limited resources. It achieves the same level of security with smaller keys, reducing computational costs. Due to its basis in scalar operations, such as point addition and point multiplication, the computational expense is less compared to elliptic curves. Equation (2) highlights these advantages of the Edwards curve, which can function

more efficiently with basic components and an addition rule, on a smaller field with fewer operations, making it ideal for resource-constrained environments.

$$x^2 + y^2 = 1 + dx^2y^2 \quad (2)$$

The Edwards curve builds a point using principles which are in contrast to other elliptic curves which utilize tangents and chords. This shows that in the following equation (3), (x_3, y_3) are known to be taken from the same curve if there are (x_1, y_1) and (x_2, y_2) in the Edward curve.

$$x_3 = \frac{(x_1y_2+x_2y_1)}{(a.(1+x_1y_1x_2y_{2))}}, y_3 = \frac{(y_1y_2-x_1x_2)}{(a.(1-x_1y_1x_2y_2))} \quad (3)$$

Correspondingly, doubling property is applied to Equation (3) to obtain point doubling as in Equation (4).

$$x_3 = \frac{(x_1y_1+x_1y_1)}{(a.(1+x_1^2y_1^2))}, y_3 = \frac{(y_1^2-x_1^2)}{(a.(1-x_1^2y_1^2))} \quad (4)$$

A Schnorr signature variant is used in the digital signature technique known as the Edwards-curve digital signature algorithm (EdDSA) designed in [24]. In comparison to ECDSA, EdDSA is less complicated and robust. A random number generator is not required for EdDSA to function. It is deterministic and distinct from ECDSA since it hashes the message in order to retrieve the seed. In ECDSA, the private key may leak or create a collision risk if the same random integer was employed to generate two distinct signatures. The Schnorr signature [25] is represented in Equation (5). Verification is carried out using Equation (6), and 'r' is a random nonce of signer.

$$(R, s) = (rG, r + H(X, R, m)x) \quad (5)$$

$$sG = R + H(X, R, m)X \quad (6)$$

HECAS (High-performance Edwards Curve Aggregate Signature) [26] is a powerful cryptographic tool that boosts the efficiency, security, and scalability of digital signatures in blockchain and other distributed systems. By using Edwards curve cryptography, known for its speed and robustness against attacks, HECAS combines multiple signatures into an aggregate one. This reduces storage needs and communication overhead, making it easier to handle large-scale operations. It is especially useful for systems like IoT, where resources are limited, as it keeps computational demands low while ensuring strong protection against tampering.

HECAS initially generates a public-private key pair as in Equation (7), where 'A' is the public key, 'G' is the generator point of the Edwards curve and 'a' is the private key.

$$A = a \,.\, G \quad (7)$$

A signature pair *(R, S)* is generated by the signer to sign the message m as in Equations (8) and (9), where 'r' is the random scalar nonce, 'H' is the cryptographic hash function and 'q' is the order of the elliptic curve.

$$R=r \cdot G \quad (8)$$

$$S=r + H\,(m, R, A) \cdot a \,(mod\ q) \quad (9)$$

When multiple signatures (R_i, S_i) are created for messages m_i they can be combined into a single signature as shown in Equations (10) and (11) where R_{agg} is the aggregated public commitment and S_{agg} is the aggregated scalar. This reduces the size and communication cost of the signatures.

$$R_{agg} = \ {}_{i=1}^{n}\Sigma\, R_i \quad (10)$$

$$S_{agg} = \ {}_{i=1}^{n}\Sigma\, S_i \quad (11)$$

The validity of an aggregated signature is verified by Equation (12) where the LHS computes the aggregated scalar multiplied by the generator G and the RHS checks the sum of all commitments and hashed values weighted by their corresponding public keys. If the equality holds, the aggregated signature is valid.

$$S_{agg}\,.\,G = \ R_{agg} + \ {}_{i=1}^{n}\Sigma\, H\,(m_i,\ R_i,\ A_i\,).\ A_i \quad (12)$$

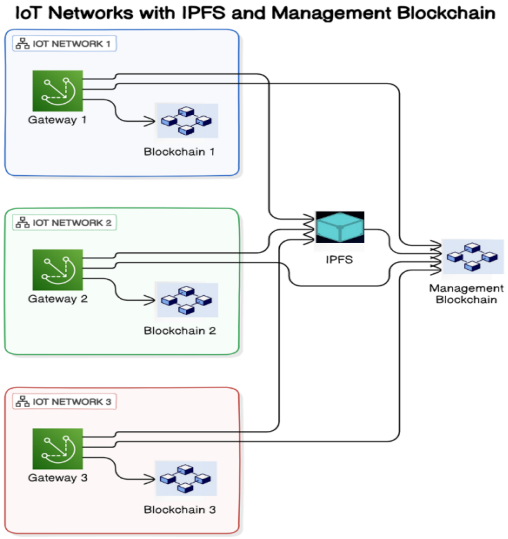
A Bloom filter [27] is a space-efficient probabilistic data structure used to test whether an element is a member of a set. It is used to determine if an element is *definitely not in the set* or *possibly in the set*, but it may produce false positives. A Bloom filter uses multiple hash functions to map an element to a fixed-size bit array, setting

specific bits to 1. To check membership, the same hash functions are applied to the element, and if all corresponding bits are 1, the element is considered *possibly* in the set; otherwise, it is definitely not. Bloom filters are commonly used in applications requiring fast lookups with minimal memory usage, such as caching, networking, and blockchain systems.

3. ARCHITECTURE

The proposed approach employs a multi-level blockchain framework to integrate several IoT networks with the management blockchain, with the objective of enhancing scalability and addressing data access control challenges as shown in figure 1. This solution enables decentralized data management primarily with IoT devices, utilizing a consensus procedure that eliminates resource-intensive computations. To send blockchain data from the IoT network to the management blockchain, all IoT nodes must provide signatures during the consensus phase. Using aggregate signatures helps minimize the signature size, which reduces the overall message size during the consensus process. The gateway controls the transit of data between decentralized storage and the management blockchain, assuring node authenticity and integrity using compressed signatures that prevent tampering and forgery.

Fig 1: Scalable IoT networks integrated with IPFS



a. Leader Election Based on Energy

In the proposed blockchain-based system, the network or cluster leader is elected based on the device with the highest residual energy to establish consensus as described in algorithm 1. Each network operates its own blockchain, and the leader plays a crucial role in validating transactions and maintaining consensus. By selecting the device with the highest energy, it is ensured that the leader has the necessary resources to perform these tasks efficiently, reducing the likelihood of interruptions due to energy depletion. This method promotes network stability and reliability, as the energy-rich leader can handle the computational demands of blockchain operations and manage network consensus without frequent changes. Consequently, this technique improves performance and durability of the blockchain network, ensuring robust and uninterrupted consensus processes. This algorithm ensures that the leader, chosen for its optimal energy resources, can effectively handle the demands of blockchain consensus and network management.

Algorithm 1: Leader Election Based on Energy
Initialize:
Define the set of devices N= $\{D_1, D_2, \ldots, D_N\}$.
Assign energy levels E(D_i) for every D_i.
Compute:
Calculate E(D_i) for all D_i.
Identify the device with the highest energy: $D_{max} = \arg\max_{D_i \in N} E(D_i)$
Where E(D_{max})= $\arg\max_{D_i \in N} E(D_i)$
While (network is active) do
For (each region) do
Update
- Elect D_{max} as the leader
- Notify all the devices about the leader
Update and analyze
-If energy of D_{max} drops below a threshold, then:
-Search for next device with highest energy
end for
end while

b. *Leader Transition and Blockchain Update with HECAS*

At the end of its operational cycle, the elected network or cluster leader communicates the current state of the blockchain to the gateway using the High-Performance Edwards Curve Aggregate Signature (HECAS) method as shown in algorithm 2. This ensures that all transactions and consensus data are authenticated, aggregated, and securely transmitted in a compact form to minimize overhead. The use of HECAS facilitates efficient and tamper-resistant synchronization of blockchain records across the network and the central management system, enhancing security and performance.

Following this handover, a new leader is elected based on the highest remaining energy among the devices. The HECAS method is also employed during the leader election process to verify and aggregate device signatures, ensuring the integrity and authenticity of the election. This rotation ensures that the leader with the most energy resources takes over, maintaining optimal performance and stability for blockchain operations.

The transition process, supported by HECAS, minimizes disruptions and ensures continuous, secure, and effective management of network consensus and data integrity throughout the system. This algorithm guarantees a smooth transition of leadership while keeping the blockchain up-to-date, maintaining network stability, and bolstering trust in the system through cryptographic assurance.

Algorithm 2: Leader Transition and Blockchain Update with HECAS
Ensure: NewLeaderElected, BlockchainUpdated
End of Cycle
if *C.cycleEnd()* = true then
- Communicate Blockchain State
- *S(C)* = getBlockchainState*(C)*
-Sign and Transmit
-$\sigma_C = Sign_{k_c}(S(C))$
- Send $(S(C), \sigma_C)$ to the gateway
end if

if Verify ($S(C), \sigma_C, k_c$) = true then
-Update Blockchain:
-updateStatus=gateway.updateBlockchain($S(C)$)
-if updateStatus = Succcess then
-BlockchainUpdated = true
end if
Collect Energy Data
For each device $D_i \in N$ do
-E**(**D_i) = energy level of device D_i
-Sign energy: $\sigma_i = Sign_{k_i}(E(D_i))$
-Aggregate signatures: $\Sigma = A\text{ggregate}(\sigma_1, \sigma_2, \ldots \sigma_n)$
Elect New Leader
if verify (Σ) = true then
-Identify new leader:
-$D_{max} = \arg\max D_i \epsilon N \;\; E(D_i)$
- Assign new leader:
- newLeader = D_{max}
- NewLeaderElected = true
end if
Transition Leadership
C = newLeader
Inform the network
For each device in *N* do
-Notify newLeader
-Update network configuration:
-NotifyAllDevices(N, newLeader)
-UpdateNetworkConfiguration(N, newLeader)

c. Blockchain Data Storage via Gateway and IPFS

At the end of each cycle, the gateway receives the updated blockchain data, which contains all the transactions and activities that occurred during that period. The gateway then transmits this data to the IPFS, where it is stored as a unique Content Identifier (CID) used to reference and retrieve content in a decentralized manner. By utilizing IPFS, the blockchain data is distributed across multiple nodes, enhancing both the scalability and resilience of the network. The data from IPFS, including CIDs or hashes, are collected by the gateway and stored on the management blockchain. To uniquely identify the data belonging to a specific network, a *NetworkID* is utilized, while a *GatewayID* is used to identify the particular gateway that processed and stored the data. By associating each CID or hash with its respective *NetworkID* and *GatewayID*, the management blockchain maintains a clear and organized record of the distributed data. The gateway also holds a mapping of the *TransactionID* to *BlockID,* While the nodes store their *TransactionIDs*. This is detailed in algorithm 3.

Algorithm 3: Blockchain Data Storage via Gateway and IPFS
Ensure: Blockchain storage
End of cycle
if cycleEnds() = true then
-Retrieve the updated blockchain data
$B_{updated} = getUpdatedBlockchainData()$
-Send the data to the gateway
$G.receive(B_{updated})$

Transmit to IPFS
-Gateway transmits the updated blockchain data to IPFS

$G.sendToIPFS(B_{updated})$

Store Data on IPFS
-IPFS stores the updated blockchain data and generates a Content Identifier (CID):

$CID=I.storeData(B_{updated})$

Retrieve CID
-Gateway receives the CID from IPFS:

receivedCID=G.receivedCIDFromIPFS(CID)

Store CID on Management Blockchain
-Store the received CID on the management blockchain via the gateway:

G.storeCIDOnManagementBlockchain(receivedCID)

Repeat Process
Wait for the next cycle to end and repeat the process:

while true:
if cycleEnds() = true
Go to step 1
end if
end while

d. Data Retrieval Using Bloom Filter in IoT Network

For access management in IoT networks, when a particular IoT device with a specific *'DeviceID'* from a *'NetworkID'* requests data, the Gateway uses a Bloom filter as an indexing method to efficiently retrieve the necessary information from the management blockchain. The procedure is given below and is detailed in algorithm 4.

When a device with a specific *DeviceID* from a particular *NetworkID* requests data, the Gateway initiates the data retrieval process. It uses a Bloom filter, a space-efficient probabilistic data structure that acts as an index to quickly determine if the data is present and also by using the *TransactionID* to *BlockID* mapping. The Bloom filter is preloaded with hashes or identifiers of stored CIDs, *NetworkIDs*, and *GatewayIDs* from the management blockchain. The Gateway hashes the *NetworkID* and *GatewayID* along with the requested data identifiers to query the Bloom filter, which indicates whether the data is likely present on the management blockchain. If the filter suggests the data exists, the Gateway retrieves the exact CIDs or hashes from the management blockchain using the *NetworkID* to identify the specific network and *GatewayID* to locate the relevant gateway data. This approach allows the Gateway to efficiently find the data without scanning the entire blockchain, making the search process faster and more resource-efficient. Once the data is confirmed, the Gateway retrieves the records and delivers the requested information to the device.

Algorithm 4: Data Retrieval Using Bloom Filter in IoT Network
Let BM represent the management blockchain.
Let BF denote the Bloom filter
Receive data request
-Receive a data request from a device in the network:

request=receiveRequestFromDevice(D_i,N), where is the requesting device, and N is the network identifier.

Initialize Bloom filter
-Initialize a Bloom filter and load it with data from the management blockchain:

BF=initializeBloomFilter()
loadBloomFilter(BF,BM)

Check with Bloom filter
-Compute the query hash and check the Bloom filter for the presence of the requested data:
queryHash=hash(N,G,requestedDataIdentifiers)
isPresent=BF.query(queryHash)
Verify data presence
-Verify whether the data is present based on the Bloom filter check:
if isPresent=true, proceed to retrieve data (Step 5)
if isPresent=false, return "Error Data not available".
Retrieve data from management blockchain
-Locate and retrieve the required data from the management blockchain:
networkData=locateNetworkData(BM,N)
gatewayRecords=findGatewayRecords(networkData,G) =
retrievedData=retrieveCIDsOrHashes(gatewayRecords,requestedDataIdentifiers)
Provide access
-Return the retrieved data to the requesting device:
returnDataToDevice(D_i,retrievedData)
Repeat process
The process repeats indefinitely:
Repeat: While true, wait for next data request, then go to Step 1

4. IMPLEMENTATION

The experiment was conducted on a desktop PC equipped with an Intel Core i5-1235U processor (1300 MHz) and 8 GB of RAM. The implementation predominantly uses Go (also known as Golang), an open-source programming language designed by Google to address challenges related to concurrency, scalability, and maintainability in large-scale software development.

Each node or IoT device in the network is implemented as a Docker container, with the nodes interconnected via a Docker network. These networks are externally linked to the decentralized storage system IPFS through a gateway. For this purpose, Kubo (previously go-ipfs), primarily developed in Go, is utilized.

All inter-node communication uses the gRPC protocol, a high-performance, open-source RPC (Remote Procedure Call) framework developed by Google. It supports efficient, cross-platform client-server communication systems and uses HTTP/2 for transport, Protocol Buffers (protobuf) for serialization, and a robust mechanism for defining services and methods in a clear, language-neutral format. Communication between the gateway and IPFS is carried out using the HTTP protocol. The local network nodes are initialized using Docker (https://www.docker.com/).

The performance evaluation results are presented in this section, comparing the PoW (Proof of Work) and PoS (Proof of Stake) consensus mechanisms. The primary goal of this evaluation is to determine the scalability potential of the IoT network model. The system's capability to connect common devices to the Internet is assessed, focusing on its ability to handle an increasing number of devices over time, including those with limited processing power. A gateway typically connects to a network either wirelessly or via a wired configuration; however, the simulation in this study uses the gRPC protocol.

The framework under consideration is specifically designed for scalable systems that leverage blockchain technology to ensure security, regardless of the communication protocol employed. The performance of the framework is evaluated through tests analyzing metrics such as throughput, latency, CPU usage, network bandwidth utilization, and storage requirements.

5. RESULTS AND DISCUSSION

This section evaluates the key attributes of the proposed framework and compares them with existing methods, highlighting the contributions to scalable IoT networks integrated with blockchain and IPFS. The experiment involves multiple IoT networks communicating commercially important data messages, secured using a robust signing mechanism like HECAS. With millions of transaction data points, the signing and verification processes

are designed to be fast and efficient. A Bloom filter is employed to enhance data retrieval from the management blockchain, enabling quick membership checks and reducing query overhead. This ensures secure, scalable, and rapid data handling in large-scale IoT systems.

a. Latency and Throughput

The performance of the IoT framework is assessed by evaluating two key metrics: *latency* and *throughput*. This process includes crucial steps such as validation, data addition to blocks, and transaction throughput measurement.

-Latency: The time required for a data packet to pass through the gateway and be appended to the blockchain.

-Throughput: The number of successfully completed transactions measured from the first to the last transaction. As the number of blockchain IoT nodes per gateway increases, the observed changes are recorded to evaluate the system's performance. The total number of blockchain IoT nodes in the experiment ranged from 60 to 500. The block size was fixed at 1 MB, and the payload size was set at 50 bytes. Figure 2 shows the block creation time for different number of IoT nodes.

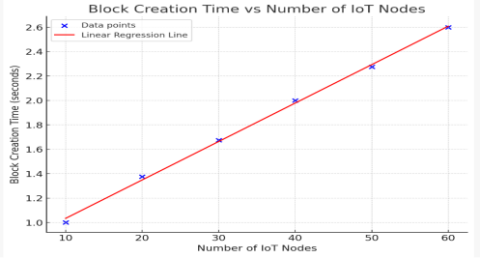
-Regression Analysis:

The relationship between the number of IoT nodes and block creation time is illustrated using a graph. A linear regression model fitted to the data shows:

-Slope: Approximately 0.0315, indicating that for every additional IoT node, the block creation time increases by about 0.0315 seconds.

-Intercept*:* Approximately 0.718, representing the base block creation time (with zero nodes) of about 0.718 seconds.

Fig 2: Comparison of Block creation time for different number of IoT nodes



To represent the relationship between the number of IoT nodes (N) and the Transactions Per Second (TPS) for the Proposed Work ($T_{Proposed}$) we derive a linear equation (13) based on the data:

$$T_{Proposed} = 75N + 1250 \quad (13)$$

Table 1 highlights the superior performance of the proposed consensus mechanism compared to traditional methods like Proof of Work (PoW) and Proof of Stake (PoS):

1. **Proof of Work (PoW)**
   - Relies on solving energy-intensive cryptographic puzzles.
   - Results in high energy consumption, long processing times, and limited scalability.
   - TPS increases by only 1 TPS for every additional 10 IoT nodes, reflecting its inefficiency.
2. **Proof of Stake (PoS)**
   - Reduces energy consumption but introduces centralization risks due to staking requirements.
   - Growth remains linear and slow, with TPS increasing by 5 TPS for every 10 additional nodes.
3. **Proposed Consensus Mechanism**
   - Eliminates the need for computational work and staking systems.

- Achieves significantly faster and more scalable transaction validation.
- TPS increases by approximately 650 TPS for every additional 10 IoT nodes, demonstrating superior scalability and processing efficiency.

Table 1: Comparison of throughput for PoW and PoS for varying number of IoT nodes

Number of IoT Nodes	Transactions per Second (TPS) - PoW	Transactions per Second (TPS) - PoS	Proposed Work (TPS)
10	5	15	2000
20	6	20	2750
30	7	25	3350
40	8	30	4000
50	9	35	4550
60	10	40	5200

b. Scalability

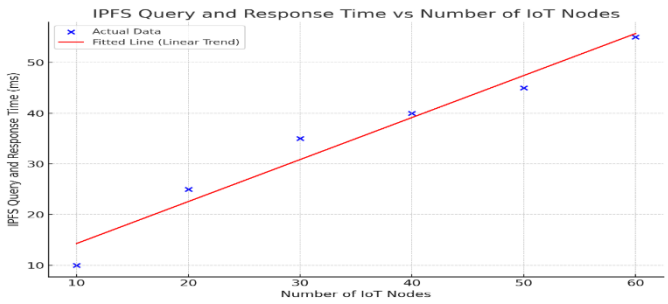
The outcomes are noticeable through the analysis of Figs. 2 and table 1, it becomes evident that the throughput and transaction speed observed within the current approach speed has good linear scalability when the number of nodes increases. This demonstrates that proposed approach outperforms the PoW and PoS approach and performs well when the number of blockchain IoT nodes increases.

c. IPFS query and response time

The time taken by the gateway node to query the IPFS and receive the required data in response is measured across multiple networks with varying numbers of IoT nodes. The figure 3 presents a graph depicting the relationship between the number of IoT nodes and the IPFS query-response time, along with a fitted linear trend line. Linear regression analysis reveals that the response time increases by approximately 0.83 ms for each additional IoT node, indicating a predominantly linear growth in response time as the network size expands and follows the equation (14). These results strongly advocate for the adoption and promotion of IPFS in distributed storage applications.

$$IPFS\ query\ and\ response\ time\ (ms) = 0.9 \times Number\ of\ IoT\ Nodes + 1 \quad (14)$$

Fig 3: IPFS query and response time for the proposed approach for varying number of IoT nodes



d. Data retrieval using Bloom's filter

The gateway processes a transaction data request from an individual IoT network node by querying the management blockchain, which holds CIDs retrieved from IPFS. During this process, a Bloom filter is populated with bit positions representing data presence. If all the queried positions are set to 1, it suggests that the data is likely available. This functionality is achieved using SHA-256 and MURMUR3 hashing functions. The measured time to confirm the presence of a block, based on its Block ID, is 1.21345 seconds, as illustrated in Fig. 4, whereas the time required to indicate the absence of a block is 0.02476 seconds, as depicted in Fig. 5.

Fig 4: Time measured for block retrieval using Bloom's filter

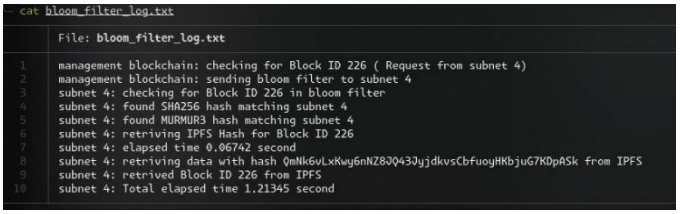
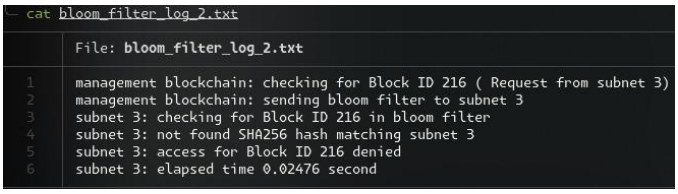


Fig 5: Time taken to notify that block is absent



From the analysis of the values and comparison with the linear search from the table 2, the logarithmic models for the times are as follows:

Table 2: Comparison of times for linear search and MURMUR3 hashing for different number of IoT nodes

No. of nodes	Time for Linear Search in msec	Time for first hash in msec	Time for second hash in msec	Total time in msec
10	13.6	7.2	7.3	14.5
20	14.39	7.6	7.8	15.4
30	15.59	8.01	7.8	15.81
50	16.45	8.1	8.05	16.15
100	18.23	8.25	8.27	16.52
150	19.49	8.3	8.32	16.52
200	20.98	8.35	8.27	16.62
250	22.45	8.34	8.3	16.64
300	24.93	8.33	8.35	16.68
350	27.22	8.35	8.29	16.64

For nodes beyond 100, the times stabilize around 8.3 ms for both the first and second hash processes. The total time can be computed as the sum of T_1 and T_2 as shown in eqns. (15), (16)

Time for first hash (T_1):

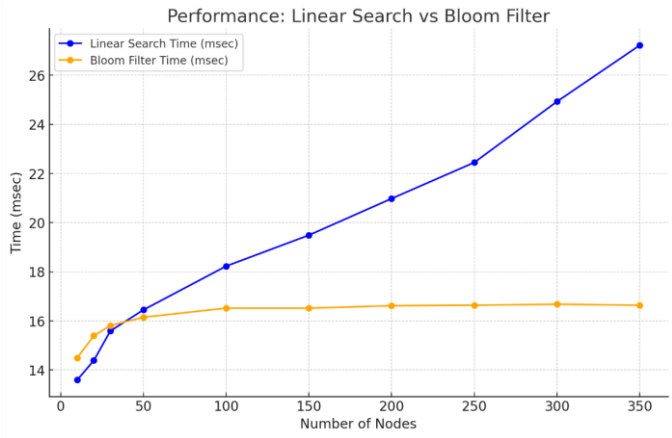
$$T_1 = \begin{cases} 6.23 + 0.46 \,.\ln(No.of\ nodes), & if\ No.of\ nodes \leq 100 \\ 8.3, & if\ No.of\ nodes > 100 \end{cases} \quad (15)$$

Time for first hash (T_2):

$$T_2 = \begin{cases} 6.46 + 0.40 \,.\ln(No.of\ nodes), & if\ No.of\ nodes \leq 100 \\ 8.3, & if\ No.of\ nodes > 100 \end{cases} \quad (16)$$

Figure 6 illustrates the comparison between Linear Search and Bloom Filter. Linear Search, with its O(n) complexity, experiences a linear increase in time as the number of nodes grows. In contrast, the time required for a Bloom Filter search remains nearly constant at O(1). Although Bloom Filters are probabilistic and can produce false positives, their efficiency makes them significantly faster and more advantageous for quick membership tests in large datasets. This efficiency gap becomes increasingly evident as the dataset size grows.

Fig 6: Comparison between Linear search and Bloom's filter for varying number of nodes



e. Signing and verification of transactions using HECAS

The **High-Performance Edwards Curve Aggregate Signature (HECAS)** scheme simplifies cryptographic processes by combining multiple digital signatures into a single, compact signature. This approach helps reduce the time and effort needed for both computation and communication, making it particularly useful in IoT networks.

1. **Signing**: Each IoT device creates its own digital signature using Edwards curve cryptography, which is both secure and efficient.
2. **Aggregation**: These individual signatures are then merged into one unified signature that can be verified all at once, instead of one at a time.

By streamlining the process, HECAS saves valuable resources and time, especially in networks with many devices, while maintaining strong security and scalability.

Here **HECAS** scheme is used, which is designed to make cryptographic operations faster and more efficient in IoT environments to reduce the time needed for signing and verification, making it highly scalable for large networks. To see how well it performs, we compared it with the **Edwards Curve Signature (ECS)** scheme, a widely used cryptographic method. The comparison focuses on how quickly each scheme handles signing and verification as the number of IoT nodes increases and is shown in table 3. The results as seen from figures 7 and 8 clearly show that HECAS is more efficient and better suited for larger networks.

Table 3: Comparison of signing and verification times of ECS and HECAS schemes

Number of IoT Nodes	Time for signing in msec (ECS)	Time for signing in msec (HECAS)	Time for verification in msec (ECS)	Time for verification in msec (HECAS)
0	0	0	0	0
20	0.8	0.81	0.9	0.8
40	1.5	1.45	1.9	1.7
60	2.2	2.1	2.9	2.3
80	3.5	3.4	3.8	3.2
100	4.8	4.1	4.8	4.1
120	6	5.2	5.9	5.06

Fig 7: Comparison of signing times for ECS and HECAS

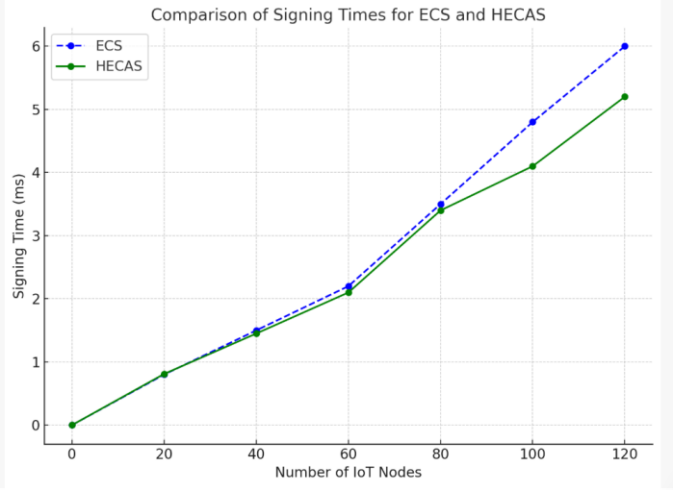
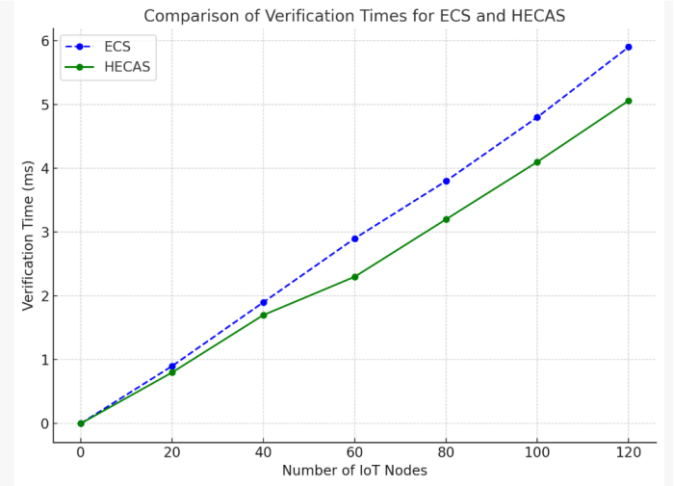


Fig 8: Comparison of verification times for ECS and HECAS



6. CONCLUSION

The proposed work introduces a methodology for data access and management in resource-constrained IoT networks. By leveraging verification and validation processes through an elected leader and gateway, the approach ensures end-to-end security while minimizing risks of device performance degradation. Performance metrics such as latency, throughput, and resource utilization are analyzed for networks ranging from 60 to 500 devices. Docker is utilized to evaluate the network's performance concerning throughput, latency, and the distributed storage provided by IPFS.

The investigation is divided into four sections: analysis of distributed storage latency, throughput, and IPFS query and response times. The proposed methodology is particularly beneficial for IoT applications requiring resource efficiency and low latency, making it suitable for real-time use cases in finance and medical sectors due to its reduced latency and enhanced throughput. Additionally, the widespread adoption of blockchain technology for secure and efficient data management and storage in medium to large enterprises depends on its low operational costs.

REFERENCES

[1] Mafei, A. A., Lavric, A., Petrariu, A. I. & Popa, V. Massive data storage solution for IoT devices using blockchain technologies. Sensors 23(3), 1570 (2023).

[2] A Holst. Iot Connected Devices Worldwide 2019–2030— Statista. https://www.statista.com/statistics/1183457/iot-connected-devic es-worldwide/ Accessed on 21-October-2022.

[3] Selvarajan, S. & Mouratidis, H. A quantum trust and consultative transaction-based blockchain cybersecurity model for healthcare systems. Sci. Rep. 13(1), 7107 (2023).

[4] Hossein Shafagh, Lukas Burkhalter, Anwar Hithnawi, and Simon Duquennoy. Towards Blockchain-Based Auditable Storage and Sharing of IoT Data. In Proceedings of the 2017 on Cloud Computing Security Workshop, CCSW '17, page 45–50, New York, NY, USA, 2017. Association for Computing Machinery.

[5] Vinothkumar, T., Sivaraju, S. S., Tangavelu, A. & Srithar, S. An energy efcient and reliable data gathering infrastructure using the Internet of Tings and smart grids. Automatika 64(4), 720–732 (2023).Ul Haque, E. et al. Cyber forensic investigation infrastructure of Pakistan: an analysis of the cyber threat landscape and readiness. IEEE Access 11, 40049–40063 (2023).

[6] Novo, O. Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Tings J. 5(2), 1184–1195 (2018).

[7] Khan, A. A., Laghari, A. A., Li, P., Dootio, M. A. & Karim, S. The collaborative role of blockchain, artifcial intelligence, and industrial internet of things in digitalization of small and medium-size enterprises. Sci. Rep. 13(1), 1656 (2023).

[8] Kutub, T., Al-Sakib, K. P. & Sadia, I. Internet of things (IoT). In Emerging ICT Technologies and Cybersecurity: From AI and ML to Other Futuristic Technologies 165–183 (Springer, 2023).

[9] Kunhahamed, P. K. & Rajak, S. Application of blockchain in mining 4.0. In Blockchain and its Applications in Industry 4.0 (eds Suyel, N. & Kemal, A.) 123–137 (Springer, Singapore, 2023).
[10] Naz, M., Al-zahrani, F. A., Khalid, R., Javaid, N., Qamar, A. M., Afzal, M. K., & Shafiq, M. (2019). A secure data sharing platform using blockchain and InterPlanetary file system. Sustainability, 11(24), 7054.
[11] Sathiya Devi, S., & Bhuvaneswari, A. (2023). Design of efficient storage and retrieval of medical records in blockchain based on InterPlanetary File System and modified bloom tree. *Security and Privacy*, *6*(5), e301.
[12] Barker, E., Roginsky, A., Davis, R., 2020. Recommendation for cryptographic key generation. National Institute of Standards and Technology, Gaithersburg, MD. doi: 10.6028/nist.sp.800-133r2.
[13] Boneh, Dan, et al. "Aggregate and verifiably encrypted signatures from bilinear maps." Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. Springer Berlin Heidelberg, 2003.
[14] Fang, Weidong, et al. "Digital signature scheme for information non-repudiation in blockchain: a state-of-the-art review." *EURASIP Journal on Wireless Communications and Networking* 2020 (2020): 1-15.
[15] Ghinea, Diana, et al. "Hybrid post-quantum signatures in hardware security keys." *International Conference on Applied Cryptography and Network Security*. Cham: Springer Nature Switzerland, 2023.
[16] Gong, Qinghua, et al. "SDACS: Blockchain-Based Secure and Dynamic Access Control Scheme for Internet of Things." Sensors 24.7 (2024): 2267.
[17] Dennis, R.; Owenson, G.; Aziz, B. A temporal blockchain: A formal analysis. In Proceedings of the 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 31 October–4 November 2016; pp. 430–437.
[18] M. Zamani, M. Movahedi, and M. Raykova, “Rapidchain: Scaling blockchain via full sharding,” in Proc. ACM CCS’18, pp. 931–948, 2018.
[19] Abdi, Adam Ibrahim, et al. "Hierarchical blockchain-based multi-chaincode access control for securing IoT systems. *Electronics* 11.5 (2022): 711.
[20] Luu, L., et al.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)
[21] KokorisKogias, E., et al.: A secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 583–598. IEEE (2018)
[22] Hamburg, Mike. "Ed448-Goldilocks, a new elliptic curve." Cryptology ePrint Archive (2015).
[23] Edwards, Harold. "A normal form for elliptic curves." Bulletin of the American mathematical society 44.3 (2007): 393-422.
[24] Josefsson, Simon, and Ilari Liusvaara. *Edwards-curve digital signature algorithm (EdDSA)*. No. rfc8032. 2017.
[25] Maxwell, Gregory, et al. "Simple schnorr multi-signatures with applications to bitcoin."Designs, Codes and Cryptography 87.9 (2019): 2139-2164.
[26] Jayabalasamy, Guruprakash, and Srinivas Koppu. "High-performance Edwards curve aggregate signature (HECAS) for nonrepudiation in IoT-based applications built on the blockchain ecosystem." Journal of King Saud University-Computer and Information Sciences 34.10 (2022): 9677-9687.
[27] Yadav, Purshotam S. "Fast and Efficient UserID Lookup in Distributed Authentication: A Probabilistic Approach Using Bloom Filters." International Journal of Computing and Engineering 6.2 (2024): 1-16.