A Blockchain Approach to IoT Sensor Data Storage Using Hyperledger Fabric

Diogo Solipa[1], José Saias[1,2][0000−0003−3025−0687], and Pedro Salgueiro[1,2][0000−0001−7614−2951]

[1] Universidade de Évora, Largo dos Colegiais 2, 7004-516, Évora, Portugal
[2] University of Évora, ALGORITMI Research Center/LASI, VISTA Lab
m51023@alunos.uevora.pt, {jsaias,pds}@uevora.pt

Abstract. The integration of IoT and blockchain technologies presents a compelling opportunity to address longstanding challenges related to data integrity, trust, and decentralization in distributed systems. blockchain is a specific implementation of distributed ledger technologies (DLTs), known for its security, trust, and decentralization. IoT, on the other hand, has become increasingly relevant, generating vast amounts of data and requiring high uptime for connected devices, which are susceptible to data tampering, privacy breaches, and single points of failure.
This paper explores the integration of blockchain with an IoT network. Particular emphasis is placed on Hyperledger Fabric, a permissioned blockchain framework tailored for enterprise-grade applications. Through a combination of architectural analysis and empirical testing, this work evaluates multiple Hyperledger Fabric configurations and performance metrics such as response time, throughput, latency, scalability.
The experimental component leverages benchmarking tools, such as Hyperledger Caliper. These tests aim to quantify trade-offs between query flexibility and system performance under varying configurations and workloads. The findings support the viability of using tailored blockchain configurations for scalable and secure IoT deployments, while highlighting areas for further optimization and future work.

Keywords: IoT, Blockchain, Distributed Systems, State Management, Decentralization, Hyperledger Fabric.

1 Introduction

The aim of this paper is to understand and study Internet of Thing (IoT) and blockchain to better understand how a system could integrate both technologies, as well as understanding the existing integrations and implementations.

Blockchain caught the eye of the public with Bitcoin [20] by being the most popular cryptocurrency over the past years. It's foundations are security, trust and decentralization. The idea of this structure is to have a network distributed throughout the various participants, with decisions happening based on a community consensus, as well as securing the system with cryptographic algorithms

and techniques to ensure the fidelity of the transactions realized. Although initially proposed and used in cryptocurrencies, blockchain technology can shape various fields such as the Finantial Sector, Healthcare, Smart Grids or Government Services [14].

A technology like blockchain is based on a ledger, which can be broadly defined as an archive, or registry, for data storage. A Distributed Ledger Technology (DLT) is defined by having a data structure and supporting protocols geographically distributed, which takes the ledger concept as a fundamental foundation. Blockchain is a subtype or specific implementation of a DLT. It inherits the security of cryptographic functions and data distribution, but it's implementations vary as section 2 further demonstrates.

IoT has become increasingly relevant in every day scenarios such as, receiving and changing the oven temperature with a smartphone connected to the a smart house network. Applications in this area will generally generate high volumes of data as well as requiring high uptimes in many of the connected devices.

An extremely important factor is the validity and trustworthiness of the transferred data. In a centralized architecture it's much harder to guarantee a user of an IoT device and application, the fidelity of the received data. Based on the growing concern with data safety and fidelity, decentralized technologies like DLTs become a solution. Every node of the network has a copy of the data, and for this reason, with the increase of participants, it becomes increasingly difficult to tamper and alter said data.

In this paper we'll present an overview of the theoretical background regarding the various components that exist in a blockchain, as well as depict a simple picture of the IoT landscape. We'll also showcase our approach highlighting what could be a possible solution for this use case, with the respective experimental results. Lastly, we'll describe our view for future work and relevant conclusions.

2 Theoretical Background

2.1 Distributed Ledger Technologies

Since ancient times, ledgers have been used to record valuable information such as financial transactions and property ownership. With technological advances, these physical records evolved into digital databases. A Distributed Ledger, however, extends this concept by replicating data across a decentralized network of participants accessible globally [23]. Each participant holds a full copy of the ledger, and any update must be reflected across all instances, secured by cryptographic hash functions and agreed upon through consensus algorithms.

2.2 Blockchain

Data centralization raises concerns over data validity, as trust is placed entirely in the controlling organization. With sufficient privileges, entities could tamper with records. Distributed systems, like blockchain, offer a decentralized alternative to mitigate this risk.

Blockchain was first introduced in 1991 by Haber and Stornetta, who proposed a cryptographic chain of blocks to timestamp documents [10]. Each block contains a *Header*—with the previous block's hash, a timestamp, a *Nonce*, and the *Merkle Root*—and a *Body*. The Merkle Root represents a binary Merkle Tree summarizing all transactions via hashes of child nodes [17]. Figure 1 illustrates this structure using a simplified four-transaction example.

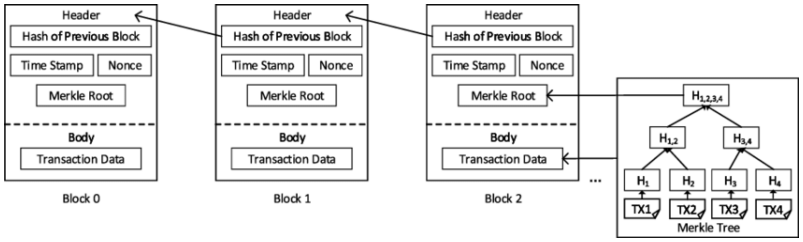

Fig. 1. Structure of a blockchain block, adapted from [17].

Blockchain, as many other DLTs, can be applied in other contexts such as health or the government [24], making it a viable option to solve the existing data security and fidelity in IoT.

Blockchain Architecture In a blockchain, a block has certain data and metadata, such as timestamps of creation, current block hash and previous hash, and any other necessary and relevant data. The first block of the chain is called the genesis block. It holds data regarding the chain and it is the first ever created, therefore, it has no previous block hash, as depicted in Figure 2.

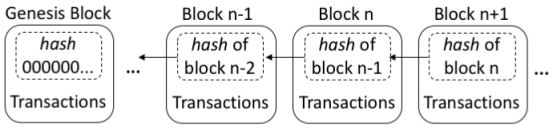

Fig. 2. Schematic view of the data structure of a blockchain [21]

New transactions are registered as a new block and included into the chain and once that is done it cannot be removed or altered [23]. A block will be accepted into the chain if it is considered valid, meaning it's been approved by the implemented consensus algorithm. Since every block has the hash of the previous block, any modified block will have all succeeding blocks recomputed which can take, in some scenarios, an impossible computational time.

Consensus Algorithms A consensus algorithm is a way for participants in a network to reach a mutual agreement on a certain addition or update to the ledger. There are several implementations of consensus algorithms, and various technologies choose to use different algorithms for their specific scenarios and use cases.

There are two big types of blockchain implementation, a Permissioned approach and a Permission-less one. The differences are rather simple; in a Private blockchain (Permissioned), for an individual or entity to become part of a network and interact with the ledger, they must be authorized to do so. A Permission-less blockchain is the exact opposite where there is no permission needed to enter or interact with the network. Figure 3 describes the various types of blockchains and the most relevant consensus algorithms they're associated with.

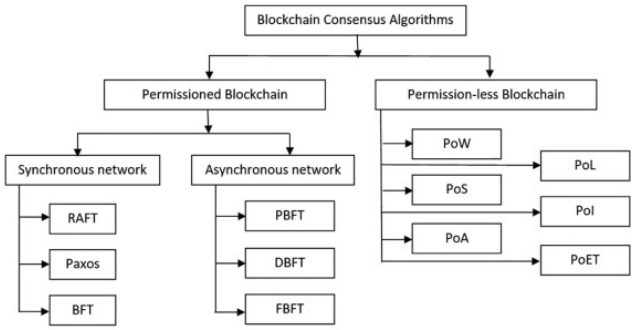

Fig. 3. Classification of Consensus Algorithms. [23]

Permission-less blockchain A Permission-less or Public blockchain is at the center of the most common and known implementations of blockchain, which is implemented in various cryptocurrencies. This type of architecture makes for a fully decentralized network across unknown participants which usually produces a larger network. With the increase in network size, decentralization broadens, which in turn enhances security against attacks, but also becomes less efficient performance wise.

Some examples of permission-less blockchains are Bitcoin [20] that uses the Proof-Of-Work (PoW) consensus algorithm [7], ethereum [13] that has merged and now uses the Proof-Of-Stake algorithm (PoS) [8]. Other examples of permission-less consensus algorithms are Proof-Of-Importance (PoI) [3], Proof-Of-Activity (PoA) [4], Proof-Of-Elapsed-Time (PoET) [4] and Proof-Of-Location (PoL) [19].

Permissioned blockchain A permissioned blockchain is a partially decentralized system where ledger access is restricted to known entities. This model en-

hances scalability and performance through controlled participation, but reduces decentralization, increasing the risk of tampering by privileged actors.

A key challenge in such networks is ensuring fault tolerance. In distributed environments, faulty nodes may lead to incorrect outcomes. This is addressed by Byzantine Fault Tolerance (BFT), where the system remains operational despite misbehaving participants [16].

As illustrated in Figure 3, consensus algorithms are typically grouped into synchronous (e.g., RAFT [11], Paxos [6]) and asynchronous models (e.g., BFT [22]). While synchronous algorithms require a common clock, this can be impractical across global systems.

Alternative protocols like PBFT [5], DBFT [26], and FBA [12] address these constraints under asynchronous assumptions.

2.3 The Hyperledger Project

The Linux foundation alongside with IBM founded the Hyperledger project, an open source initiative to build and develop an open global ecosystem for enterprise grade blockchain technologies.

Hyperledger Fabric is a highly scalable permissioned network, fully designed for enterprise usage.

Hyperledger Fabric Fabric is a modular, open-source system for building and developing Private Permissioned blockchain solutions [2]. It's modular architecture enables for various entrepise grade use cases, such as consensus mechanisms, privacy and membership services.

Fabric's architecture can be defined as ***execute-order-validate***,

- **Execute**: According to the *chaincode*, the transactions are executed, if they are endorsed (validated);
- **Order**: Introduces order between transactions, similar to sorting timestamps; follows a specific pre-defined consensus protocol;
- **Validate**: Ensures the validity of a transaction by looking into it's ordering, preventing concurrency issues.

In Fabric, *Fault handling*, any client is considered to be potentially malicious or *Byzantine*.

Fabric is written in the GO language and uses gRPC (Google Remote Procedure Call) [1], to communicate between clients, peer nodes and network orderers. Consisting of various components such as peer nodes, ordering service, membership and chaincode.

Hyperledger Fabric is extremely relevant and used as a BaaS (blockchain as a Service) [25] since it allows for minimal development to get a working prototype for decentralized data storage and distributed computing.

2.4 Internet Of Things

IoT revolutionized the way society goes about its daily tasks. Smart homes [27], smart cities, health monitoring and improvement technologies, smart industries and so many other quality of life transformations, are enabled through IoT [15].

Having devices constantly connected to the internet, makes for a whole world of possibilities regarding monitoring, optimizations and overall control over our technological lives. Tackling issues from business development and management to simple day-to-day activities and habits. Although there are many applications and development in this area, it still has a lot of investigation to be made.

Regarding development, one of the biggest issues associated with IoT is the performance and scalability of a system. Most devices will be either mobile devices or integrated systems, where the processing power is lacking, battery life can also be an issue, and having full fledged implementations will be mostly impossible [9].

IoT and Blockchain Integration Distributed Ledger systems naturally have a high interest in regards to IoT systems. Decentralizing data in a system can be a requirement or purely a positive advantage, in which IoT devices can exchange information over the internet through a secure network. Data can be easily traced, thanks to its immutability, and any confidence concerns are automatically removed from the equation. With DLTs IoT has the possibility to be completely revolutionized in many crucial areas where data integrity is key, such as health, industries, businesses, finance, and many other areas [18].

IoT can benefit greatly from DLTs, such as blockchain, on various areas like the following:

- **Decentralization**: Blockchain has the intrinsic property of decentralizing information with its peer-to-peer network. It also improves against fault tolerance, like the Byzantine Fault, as well as help with IoT scalability;
- **Identity and Autonomy**: With a standard blockchain network, participants (users) can be easily identified and its transactions easily traced. IoT systems will also have the capability of interacting without the need for servers;
- **Data Authenticity and Security**: Every piece of data that was transacted, can be traced back to their original state, validated by smart contracts. Current existing protocols in IoT can greatly benefit from this.

3 Proposed Approach

This section presents the methodology adopted to achieve the objective of integrating blockchain technology with an IoT system for secure and scalable sensor data storage. The approach is structured into three major components: (i) the conceptual design of the proposed architecture, (ii) the implementation strategy used to build the blockchain-based system, and (iii) the benchmarking methodology employed to evaluate performance and scalability under realistic IoT conditions.

3.1 Architectural Design

The proposed architecture is designed to address the key requirements of IoT data systems, namely: high throughput, tamper-resistance, decentralized trust, and efficient query capability. The core of the solution is built on *Hyperledger Fabric*.

The architectural components are as illustrated and discussed, including:

- **Fog/Edge Layer of Sensor Devices**: Simulated clients generating data;
- **ETL Layer**: Filter raw Sensor data;
- **Hyperledger Fabric Network**: An instance of HLF with specific chaincode deployed;
- **Fabric Gateway Rest Adapter**: Server running a REST wrapper of the Fabric Gateway SDK;
- **Analytics**: Analytics module for future work (LLMs, ML, etc)
- **Visualization & Monitoring**: Grafana & Prometheus for on-chain data visualization as well as network resources monitoring.

Since we couldn't replicate data streaming from a set of IoT devices on a Network, we've managed an adaptation with representative batch files. Figure 4 illustrates our approach to a fully modular implementation of IoT with blockchain for on-chain traceability and querying, whilst also being prepared for off-chain analytics.

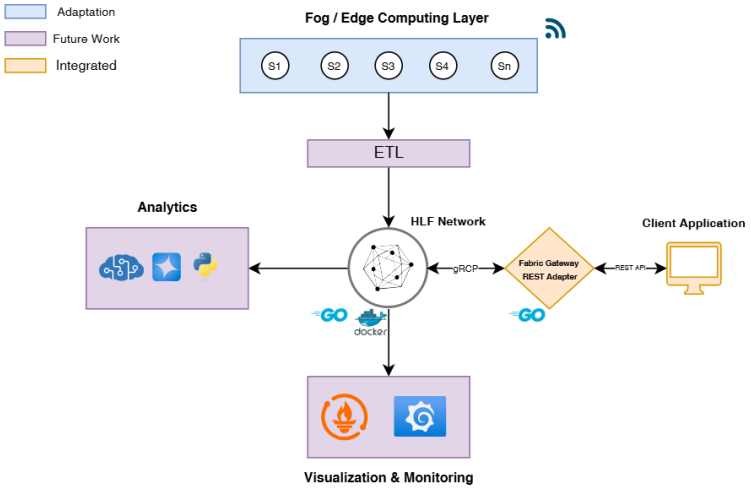

Fig. 4. High Level Design of the integration of IoT with Hyperledger Fabric

3.2 System Implementation

Following the architectural design, the blockchain network was implemented using *Hyperledger Fabric v2.4*, with a specific focus on representing sensor data as blockchain assets. The implementation consisted of two main phases: a baseline deployment using standard chaincode for calibration, and a custom deployment designed for IoT-specific data models.

Custom Chaincode was developed in Go, implementing operations for:

- Asset initialization (simulating sensor data insertions),
- Asset updates (value changes over time),
- Historical queries (`getHistoryForKey`),
- Attribute-based rich queries (with CouchDB).

Data Schema: Each sensor reading was modeled with fields such as `sensorID`, `timestamp`, `value`, and optionally `type` or `location`. Two strategies were compared:

1. **Flat Model** — each reading as a unique asset (`sensorID_timestamp` as key),
2. **Append Model** — grouping readings under one asset (array).

The resulting smart contracts allowed interaction with the blockchain network in a way that closely mimicked real-world IoT devices transmitting frequent, structured measurements.

3.3 Benchmarking Strategy

To validate the proposed system under realistic workloads, a benchmarking approach was devised combining standardized and custom performance measurements.

Tools: The primary tool used was *Hyperledger Caliper*, allowing consistent and reproducible tests of throughput, latency, and failure rates.

Test Scenarios

- Asset creation (`init`),
- Direct queries (`readByKey`),
- History retrieval (`getHistoryForMarble`),
- State mutation (`transfer`),
- Rich queries (CouchDB-specific selectors).

Metrics Captured:

- Transaction Throughput (TPS),
- Average and Maximum Latency,
- Success/Failure Rate

Additional manual testing was performed to evaluate performance under data growth and chaincode-level bottlenecks (e.g., gRPC message limits, chaincode logic complexity).

This benchmarking framework provided both a standardized evaluation (via Caliper) and an exploratory environment for stress testing and iterative optimization of the system.

4 Experimental Results

To evaluate the feasibility and performance of blockchain integration in IoT environments, we designed and executed a series of experiments. This section outlines the test setup, scenarios, and results. Key performance metrics—Throughput, Latency, TPS, and transaction success—were measured to assess how well the blockchain infrastructure supports IoT constraints.

Our experiments used Hyperledger Fabric, a modular, permissioned DLT framework suited for enterprise and IoT use cases. Fabric's architecture allows flexibility in components such as the state database. We tested both LevelDB, an embedded key-value store, and CouchDB, a document store supporting rich JSON queries, under various workloads to ensure result consistency. Hyperledger Caliper was used as the benchmarking tool to systematically measure performance across key KPIs.

As a baseline, we used the official `marbles` benchmark included with Caliper, which supports both key-based and rich-query scenarios. This enabled comparison of Fabric's behavior with LevelDB and CouchDB under uniform conditions.

After this stage, we implemented a custom Fabric network tailored for IoT, including specific chaincode and asset models to simulate real-world transactions. We also developed a Fabric Gateway client to emulate inbound requests to the blockchain. The impact of these choices is analyzed in the subsequent sections.

4.1 Benchmarking with Hyperledger Caliper

Hyperledger Caliper is the official benchmarking framework for Hyperledger technologies. It allows the execution of predefined workloads to evaluate performance under controlled and repeatable conditions.

Test Scenario: Marbles Use Case This well-known sample application in the Fabric ecosystem was selected for its simplicity and traceability. Tests were implemented using both LevelDB and CouchDB and executed various operations to benchmark key interaction patterns.

Baseline Benchmark Configuration The Caliper benchmark was structured into multiple transaction rounds, each targeting a specific access or modification pattern:

- **init**: Asset creation using `initMarble`, with 500 marbles inserted at a fixed rate of 25 TPS.
- **read-by-key**: Lookup operations using `readMarble`, simulating point queries for IoT devices.
- **getHistory**: Historical reads using `getHistoryForMarble`, retrieving state transitions for a given marble.
- **transfer**: State updates using `transferMarble`, representing value mutation over time.

All tests were executed with 5 concurrent workers under a fixed-rate strategy. Results were measured in terms of throughput (TPS), latency (min/avg/max), and transaction success rate.

Benchmarking Hyperledger Fabric with LevelDB Using Caliper To assess the baseline performance of Hyperledger Fabric under a key-value storage configuration, we conducted a series of benchmark tests using **LevelDB**, the default state database. The evaluation utilized the official `marbles` chaincode provided by Hyperledger Caliper. This allowed us to isolate Fabric's native performance characteristics in a controlled, repeatable environment before introducing rich-query capabilities via CouchDB. Table 1 represents the baseline results of the marbles benchmark using LevelDB.

Round	TPS	Avg Latency	Max Latency	Failures
`init`	25.2	0.15 s	0.27 s	0
`read-by-key`	10.5	0.01 s	0.02 s	0
`getHistory`	5.3	0.01 s	0.02 s	0
`transfer`	9.3	0.34 s	2.05 s	0

Table 1. Performance results for LevelDB-based Caliper benchmark.

Results Summary (LevelDB) The results confirm that LevelDB offers excellent performance for direct key access and high-throughput asset creation. Historical access using `getHistoryForMarble` was also performant, although initially limited in insight due to low state depth per asset. Transfer operations introduced slightly higher latency, which is expected due to endorsement and commit delays, yet remained stable without failures.

Benchmarking Hyperledger Fabric with CouchDB Using Caliper To assess the performance implications of enabling rich-query capabilities in Hyperledger Fabric, we conducted a series of benchmark tests using **CouchDB** as the state database. CouchDB supports JSON document storage and allows expressive querying using selectors. While this introduces added flexibility, it also increases the computational and I/O overhead during ledger operations.

The evaluation was conducted using the official `marbles` chaincode provided by Hyperledger Caliper. Tests mirrored those run in the LevelDB phase to enable fair comparison. All rounds were executed with 5 concurrent workers using a fixed-rate strategy. Table 1 represents the baseline results of the marbles benchmark using CouchDB.

Round	TPS	Avg Latency	Max Latency	Failures
`init`	25.2	0.18 s	0.29 s	0
`read-by-key`	10.4	0.12 s	0.14 s	0
`getHistory`	5.3	0.01 s	0.02 s	0
`transfer`	9.3	0.36 s	2.07 s	0
`query-rich`	5.3	0.02 s	0.06 s	0

Table 2. Performance results for CouchDB-based Caliper benchmark.

Results Summary (CouchDB) The results show that CouchDB supports baseline operations effectively, with only modest increases in latency when compared to LevelDB. The average latency for asset creation rose from 0.15s to 0.18s, and key lookups exhibited a minor increase from 0.01s to 0.12s. Rich query execution using JSON selectors was successful after correcting the input format, highlighting the need for precise query string handling. Historical queries and transfer operations remained stable, indicating that CouchDB's additional indexing complexity does not significantly affect these interactions at low concurrency levels. It's also important to highlight that the performance of `getHistory` is very similar as this is a shallow read, meaning that every asset has never been transfered, and therefore has no dependecies.

Comparison: LevelDB vs. CouchDB Performance LevelDB delivered consistently lower latency in asset creation and key lookups, making it well-suited for high-throughput IoT scenarios with simple queries. Historical reads and state updates also performed efficiently under this backend.

CouchDB introduced slightly higher latency but enabled rich queries using JSON selectors. Despite the added flexibility, basic operations remained stable, and historical/query performance was comparable under light workloads.

In summary, LevelDB favors performance, while CouchDB offers greater query expressiveness with moderate overhead.

4.2 Manual Testing with IoT-Oriented Data Structures

Recognizing Caliper's limitations in modeling dynamic or use-case-specific scenarios, a set of custom benchmarks was developed to simulate realistic IoT workloads. These involved direct interaction with chaincode using custom data schemas. To interact with the Hyperledger network we created a client that sent

and received requestes from the Fabric Gateway, responsible for managing every inbound and outbound requests.

Dataset Characteristics

- Source: Real-world IoT dataset
- Size: 100,000 entries
- Test sample sizes: 10, 1,000, and 100,000 records

Data Models Evaluated

- **Append-to-List Model:** Each sensor has a single ledger asset, and new data points are appended to a list.
  - Limitation: Rapid asset growth degrades performance significantly.
  - Limitation: Exceeds default gRPC message size (4MB) after ∼14,769 entries.
- **Sensor and Timestamp Key:**
  - Use `sensorID:timestamp` as a key for each reading.
  - Enable efficient range queries (especially in CouchDB).

5 Conclusion

Over this paper we presented our approach and benefits to integrating IoT ecosystems into a blockchain network, ensuring safety, transparency and efficiency. Using permissioned blockchains we're able to bypass some of the performance constraints in comparison to public examples such as Bitcoin. Bulding such solutions carries a highly complex development challenge, that can make its corporate usage limited, and consequent mass adoption. For the future of our work, it's important to fully test the proposed data structure under various loads, with multiple clients and peers. We envision the creation of various modules as depicted in the HLD, focusing on the importance of an ETL layer for quality data processing and a Visualization tool for a high level monitoring of the system. Furthermore, and with the relevance and ease of use of AI, we propose the development of an Analytics module, that could not only infer from the blockchain by indexing it in a Vectorial DB, but also with standard Machine Learning models for predictive analysis and forecasting.

References

1. Grpc, https://grpc.io/
2. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S.W., Yellick, J.: Hyperledger fabric: A distributed operating system for permissioned blockchains. CoRR **abs/1801.10228** (2018), http://arxiv.org/abs/1801.10228

3. Auhl, Z., Chilamkurti, N., Alhadad, R., Heyne, W.: A comparative study of consensus mechanisms in blockchain for iot networks. Electronics **11**(17) (2022). https://doi.org/10.3390/electronics11172694, https://www.mdpi.com/2079-9292/11/17/2694
4. Bach, L.M., Mihaljevic, B., Zagar, M.: Comparative analysis of blockchain consensus algorithms. In: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). pp. 1545–1550 (2018). https://doi.org/10.23919/MIPRO.2018.8400278
5. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (Nov 2002). https://doi.org/10.1145/571637.571640
6. Charapko, A., Ailijiang, A., Demirbas, M.: Bridging paxos and blockchain consensus. In: iThings/GreenCom/CPSCom/SmartData [6], pp. 1545–1552, http://dblp.uni-trier.de/db/conf/ithings/ithings2018.htmlCharapkoAD18
7. Cheng, Z., Wu, G., Wu, H., Zhao, M., Zhao, L., Cai, Q.: Deterministic proof of work. CoRR **abs/1808.04142** (2018), http://dblp.uni-trier.de/db/journals/corr/corr1808.htmlabs-1808-04142
8. Gazi, P., Kiayias, A., Russell, A.: Stake-bleeding attacks on proof-of-stake blockchains. In: CVCBT [8], pp. 85–92, http://dblp.uni-trier.de/db/conf/cvcbt/cvcbt2018.htmlGaziKR18
9. Gurbani, R.: Iot: Ongoing challenges and opportunities in mobile technology. International Journal of Electrical, Electronics and Computers **6**, 1–9 (01 2021). https://doi.org/10.22161/eec.63.1
10. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. J. Cryptol. **3**(2), 99–111 (1991), http://dblp.uni-trier.de/db/journals/joc/joc3.htmlHaberS91
11. Huang, D., Ma, X., Zhang, S.: Performance analysis of the raft consensus algorithm for private blockchains. IEEE Trans. Syst. Man Cybern. Syst. **50**(1), 172–181 (2020), http://dblp.uni-trier.de/db/journals/tsmc/tsmc50.htmlHuangMZ20
12. Innerbichler, J., Damjanovic-Behrendt, V.: Federated byzantine agreement to ensure trustworthiness of digital manufacturing platforms. In: CRYBLOCK@MobiSys [12], pp. 111–116, http://dblp.uni-trier.de/db/conf/mobisys/cryblock2018.htmlInnerbichlerD18
13. Kim, S.K., Ma, Z., Murali, S., Mason, J., Miller, A., Bailey, M.: Measuring ethereum network peers. In: Internet Measurement Conference [13], pp. 91–104, http://dblp.uni-trier.de/db/conf/imc/imc2018.htmlKimMMMMB18
14. Krichen, M., Ammi, M., Mihoub, A., Almutiq, M.: Blockchain for modern applications: A survey. Sensors **22**, 5274 (07 2022). https://doi.org/10.3390/s22145274
15. Kumar, S., Tiwari, P., Zymbler, M.: Internet of things is a revolutionary approach for future technology enhancement: a review. Journal of Big Data **6** (12 2019). https://doi.org/10.1186/s40537-019-0268-2
16. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) **4**(3), 382–401 (1982)
17. Liang, Y.C.: Blockchain for Dynamic Spectrum Management, pp. 121–146 (01 2020). https://doi.org/10.1007/978-981-15-0776-2_5
18. Mehannaoui, R., Mouss, K.N., Aksa, K.: Iot-based food traceability system: Architecture, technologies, applications, and future trends. Food Control **145**, 109409 (2023). https://doi.org/https://doi.org/10.1016/j.foodcont.2022.109409, https://www.sciencedirect.com/science/article/pii/S0956713522006028
19. Migliorini, S.: Enhancing blockchain smart-contracts with proof-of-location (2018)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)

21. Tuler de Oliveira, M., Carrara, G., Fernandes, N., Albuquerque, C., Carrano, R., Medeiros, D., Menezes, D.: Towards a performance evaluation of private blockchain frameworks using a realistic workload. pp. 180–187 (02 2019). https://doi.org/10.1109/ICIN.2019.8685888
22. Qin, H., Cheng, Y., Ma, X., Li, F., Abawajy, J.: Weighted byzantine fault tolerance consensus algorithm for enhancing consortium blockchain efficiency and security. Journal of King Saud University - Computer and Information Sciences **34**(10, Part A), 8370–8379 (2022). https://doi.org/https://doi.org/10.1016/j.jksuci.2022.08.017, https://www.sciencedirect.com/science/article/pii/S1319157822002919
23. Shrimali, B., Patel, H.B.: Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities. Journal of King Saud University - Computer and Information Sciences **34**(9), 6793–6807 (2022). https://doi.org/https://doi.org/10.1016/j.jksuci.2021.08.005, https://www.sciencedirect.com/science/article/pii/S131915782100207X
24. Soltani, R., Zaman, M., Joshi, R., Sampalli, S.: Distributed ledger technologies and their applications: A review. Applied Sciences **12**(15) (2022). https://doi.org/10.3390/app12157898, https://www.mdpi.com/2076-3417/12/15/7898
25. Song, J., Zhang, P., Alkubati, M., Bao, Y., Yu, G.: Research advances on blockchain-as-a-service: architectures, applications and challenges. Digital Communications and Networks **8**(4), 466–475 (2022). https://doi.org/https://doi.org/10.1016/j.dcan.2021.02.001, https://www.sciencedirect.com/science/article/pii/S2352864821000092
26. Yusoff, J., Mohamad, Z., Anuar, M.: A review: Consensus algorithms on blockchain. Journal of Computer and Communication (3), 37–50 (2022). https://doi.org/https://doi.org/10.4236/jcc.2022.109003
27. Zhou, J., Cao, Z., Dong, X., Vasilakos, A.V.: Security and privacy for cloud-based iot: Challenges. IEEE Communications Magazine **55**(1), 26–33 (2017). https://doi.org/10.1109/MCOM.2017.1600363CM